**6 Directions for Java Performance Optimization You Should Know**



Photo by [Annie Spratt](https://unsplash.com/@anniespratt?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com/?utm_source=medium&utm_medium=referral)

In the last article “**Performance Optimization Theory**”, I introduced various indicators of performance analysis, so that when doing [performance optimization](https://medium.com/javarevisited/7-best-courses-to-learn-jvm-garbage-collection-and-performance-tuning-for-experienced-java-331705180686), there are specific optimization goals and measurement methods, and the optimization effect will not just stay in the intuitive sense. superior.

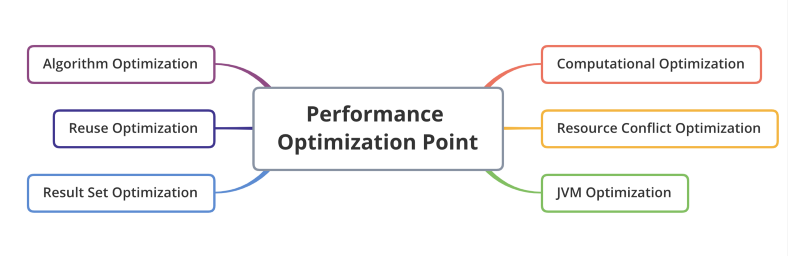
**[Performance Optimization Theory You Should Know as a Programmer](https://levelup.gitconnected.com/performance-optimization-theory-you-should-know-as-a-programmer-524119882e7b" \t "_blank)**

[Many people have only a very shallow understanding of performance optimization because they usually only focus on…](https://levelup.gitconnected.com/performance-optimization-theory-you-should-know-as-a-programmer-524119882e7b" \t "_blank)

[levelup.gitconnected.com](https://levelup.gitconnected.com/performance-optimization-theory-you-should-know-as-a-programmer-524119882e7b" \t "_blank)

Now that you know your optimization goals, what should you do next?

As programmers, in our daily work, we optimize the way, mainly technical means. This series of technical means can be roughly classified into 6 categories.

[[](https://javarevisited.blogspot.com/2019/04/top-5-courses-to-learn-jvm-internals.html)](https://javarevisited.blogspot.com/2019/04/top-5-courses-to-learn-jvm-internals.html)

It can be seen that the optimization method focuses on the planning of computing resources and storage resources. There are many ways of exchanging space for time in optimization methods, but it is not advisable to only take care of calculation speed without considering complexity and space issues. What we need to do is to achieve the optimal state of resource utilization under the premise of taking care of performance.

**1 Reuse optimization**

When writing code, you will find that there is a lot of repetitive code that can be extracted and made into public methods. In this way, the next time you use it, you don’t have to write it again.

This idea is reused. The above description is an optimization of coding logic, and for data access, there is the same multiplexing situation. Whether in life or [coding](https://medium.com/javarevisited/7-best-courses-to-learn-recursion-for-programming-and-coding-interviews-1f8b0bcfd44), repetitive things happen all the time. If there is no reuse, work and life will be more tiring.

In software systems, when it comes to data multiplexing, the first thing that comes to our mind is buffering and caching. Pay attention to the difference between these two words, their meanings are completely different, many people are easily confused, I will briefly introduce them here.

**Buffer** is commonly used for the temporary storage of data and then batch transmission or writing. The sequential method is used to alleviate frequent and slow random writes between different devices. The buffer is mainly for writing operations.

**Cache** is commonly used for multiplexing of reading data, by caching them in a relatively high-speed area, the cache is mainly for reading operations.

In Java, [database connection pools](https://javarevisited.blogspot.com/2012/06/jdbc-database-connection-pool-in-spring.html), [thread pools](https://javarevisited.blogspot.com/2013/07/how-to-create-thread-pools-in-java-executors-framework-example-tutorial.html), etc. are used very frequently. Since the cost of creating and destroying these objects is relatively high, we will temporarily store them after use. The next time we use them, we don’t need to go through the time-consuming initialization operation again.

**2 Computational optimization**

**2.1 Parallel execution**

Today’s CPUs are developing very fast, and most hardware is multi-core. To speed up the execution of a task, the fastest and optimal solution is to have it executed in parallel. There are the following three modes of parallel execution.

The first mode is multi-machine, which uses load balancing to split traffic or large calculations into multiple parts and process them at the same time. For example, Hadoop uses MapReduce to break up tasks and perform calculations on multiple machines at the same time.

The second mode is to use multiple processes. For example, Nginx adopting the NIO programming model. The Master manages the Worker process in a unified manner, and then the Worker process performs the real request proxy, which can also make good use of multiple CPUs of the hardware.

The third mode is to use multithreading, which is also the most exposed to [Java programmers](https://medium.com/javarevisited/the-java-programmer-roadmap-f9db163ef2c2). For example, Netty, which uses the Reactor programming model, also uses NIO, but it is thread-based. The Boss thread is used to receive the request and then dispatch it to the corresponding Worker thread for real business computation.

**2.2 Change from synchronous to asynchronous**

Synchronous to asynchronous, which usually involves a change in the programming model. In synchronous mode, the request will block until a success or failure result is returned. Although its programming model is simple, it is particularly problematic when dealing with sudden and skewed traffic, and requests can easily fail.

Asynchronous operations can easily support horizontal expansion, and can also relieve transient pressure and make requests smoother.

**2.3 Lazy loading**

The last one is to use some common design patterns to optimize business and improve the experience, such as **singleton patterns**, **proxy patterns**, etc. For example, when drawing a Swing window, if you want to display more pictures, you can load a placeholder first, and then slowly load the required resources through the background thread, which can avoid the window from being frozen.

**3 Result set optimization**

Next, we will introduce the optimization of the result set. To give a more intuitive example, we all know that the representation of XML is very good, so why is there [JSON](https://javarevisited.blogspot.com/2013/02/how-to-convert-json-string-to-java-object-jackson-example-tutorial.html)? In addition to being simpler to write, an important reason is that its volume has become smaller, and the transmission efficiency and parsing efficiency have become higher. Like Google’s [Protobuf](https://javarevisited.blogspot.com/2020/04/top-5-courses-to-learn-grpc-and-google-protocol-buffer-java-python-golang.html), the volume is smaller. Although the readability is reduced, in some high concurrency scenarios (such as RPC), the efficiency can be significantly improved, which is a typical optimization of the result set.

Like Nginx, GZIP compression is generally turned on to keep the transmitted content compact. The client only needs a small amount of computing power, and it can be easily decompressed. Since this operation is decentralized, the performance penalty is fixed.

Understanding this truth, we can see the general idea of ​​result set optimization, you should try to keep the returned data as concise as possible. Some fields that the client does not need, then in the code, or directly in the SQL query, remove it.

Some businesses that do not have high requirements for timeliness, but have high requirements for processing power. We need to learn from the experience of the buffer, minimize the interaction of network connections, and use batch processing to increase the processing speed.

The result set is likely to be used twice, and you may cache it, but it still lacks speed. At this time, it is necessary to optimize the processing of the data collection, using indexes or Bitmap bitmaps to speed up data access.

**4 Resource conflict optimization**

In our usual development, we will involve a lot of shared resources. Some of these shared resources are stand-alone, such as a [HashMap](https://www.java67.com/2017/08/top-10-java-hashmap-interview-questions.html); some are external storage, such as a database row; some are single resources, such as setnx of a key in [Redis](https://javarevisited.blogspot.com/2022/02/top-5-courses-to-learn-redis.html); some are coordination of multiple resources, such as transactions, distributed transactions, etc.

In reality, there are many performance issues related to locks. Most of us think of database row locks, table locks, various locks in [Java](https://medium.com/javarevisited/10-best-places-to-learn-java-online-for-free-ce5e713ab5b2), etc. At a lower level, such as CPU command-level locks, JVM instruction-level locks, operating system internal locks, etc., it can be said to be everywhere.

Only [concurrency](https://medium.com/javarevisited/6-multithreading-and-concurrency-books-every-java-programmer-should-read-b6a08d2aae54)can generate resource conflicts. That is, at the same time, only one processing request can obtain the shared resource. The way to resolve resource conflicts is to lock. Another example is a transaction, which is essentially a lock.

According to the lock level, locks can be divided into optimistic locks and pessimistic locks. Optimistic locks are more efficient; according to lock types, locks are divided into fair locks and unfair locks. There are some subtleties in the scheduling of tasks. difference.

Competition for resources will cause serious performance problems, so there will be some research on lock-free queues, which will significantly improve performance.

**5 Algorithm optimization**

Algorithms can significantly improve the performance of the complex business, but in actual business, they are often variants. As storage is getting cheaper and cheaper, in some CPU-intensive businesses, space is often used for time to speed up processing.

Algorithms belong to code tuning. Code tuning involves a lot of coding skills and requires users to be very familiar with the API of the language used. Sometimes, the flexible use of [algorithms and data structures](https://medium.com/javarevisited/7-best-places-to-practice-coding-problems-and-learn-data-structures-and-algorithms-for-interviews-646f7fe41350) is also an important part of code optimization. For example, commonly used ways to reduce time complexity include recursion, bisection, sorting, dynamic programming, etc.

A good implementation has a very large impact on the system than a poor implementation. For example, the implementation of List, LinkedList and ArrayList are several orders of magnitude worse in random access performance; another example CopyOnWriteList adopts the method of copy-on-write, which can significantly reduce the lock conflict in the scenario of reading more and writing less. When to use synchronization and when to be thread-safe, also have higher requirements on our coding ability.

In this part of the knowledge, we need to pay attention to accumulation in our everyday work.

**6 JVM optimization**

Because Java runs on a JVM virtual machine, many of its features are restricted by the JVM. Optimizing the JVM virtual machine can also improve the performance of JAVA programs to a certain extent. If the parameters are not configured properly, it may even cause serious consequences such as OOM.

JVM performance tuning involves various trade-offs, often affecting the whole body, and it is necessary to comprehensively consider the impact of all aspects. Therefore, it is very important to understand some of the internal operating principles of the JVM. It will help us deepen our understanding of the code and help us write more efficient code.