

# JVM

- [【JVM】JPS教學](#)
- [【JVM】jstat 教學](#)

# 【JVM】JPS教學

```
tomcat@aut-liveapi221:/usr/lib/jvm/java-17-openjdk-amd64/bin$ jps -q
3481575
2307111
tomcat@aut-liveapi221:/usr/lib/jvm/java-17-openjdk-amd64/bin$ jps -l
2307111 build/libs/r...-liveapi-2411.1.0.jar
3481594 jdk.jcmd/sun.tools.jps.Jps
tomcat@aut-liveapi221:/usr/lib/jvm/java-17-openjdk-amd64/bin$ jps -m
2307111 ...-liveapi-2411.1.0.jar --spring.profiles.active=uat
3481614 Jps -m
tomcat@aut-liveapi221:/usr/lib/jvm/java-17-openjdk-amd64/bin$ jps -v
3481633 Jps -Dapplication.home=/usr/lib/jvm/java-17-openjdk-amd64 -Xms8m -Djdk.module.main=jdk.jcmd
2307111 ...-liveapi-2411.1.0.jar
tomcat@aut-liveapi221:/usr/lib/jvm/java-17-openjdk-amd64/bin$ jps
3481653 Jps
2307111 ...-liveapi-2411.1.0.jar
```

`jps` 是 Java 提供的一個工具，用於列出正在執行的 Java 進程。這個工具隨 Java 開發工具包 (JDK) 一起提供，通常用於監控和診斷 Java 應用程序。以下是 `jps` 的使用方式和一些常見選項。

## 基本用法

直接執行 `jps` 命令可以顯示當前用戶下正在執行的 Java 進程的簡單列表：

```
jps
```

這會輸出一個進程 ID 和 Java 類名，例如：

```
12345 MyApplication
67890 AnotherApp
```

## 常見選項

- `-l`：顯示完整的類名或 JAR 路徑。

```
jps -l
```

輸出示例：

```
12345 com.example.MyApplication
67890 /path/to/AnotherApp.jar
```

- `-v`：顯示啟動 Java 進程時的 JVM 參數。

```
jps -v
```

輸出示例：

```
12345 MyApplication -Xms256m -Xmx512m
67890 AnotherApp -Dproperty=value
```

- `-m`：顯示傳遞給主類的參數。

```
jps -m
```

輸出示例：

```
12345 MyApplication arg1 arg2
```

## 查看特定主機的進程

如果需要查看其他主機上的 Java 進程，可以使用 `jps` 的 `hostid` 參數指定主機 ID，例如：

```
jps <hostid>
```

**注意：**此方法通常需要配置遠程連接。

# 常見用途

- 檢查是否有指定的 Java 應用在執行。
- 確認進程 ID (PID) 以便進行進一步的診斷（例如 `jstack` 或 `jmap` 等工具）。
- 快速查看進程的 JVM 參數或主程序參數。

## 小結

`jps` 是一個簡單但實用的工具，用於 Java 進程的快速概覽。配合其他 JVM 工具（如 `jstack`, `jmap` 等）使用，可以更有效地監控和診斷 Java 應用程序。

# 【JVM】jstat 教學

jstat 是 Java 提供的另一個工具，用於監控 Java 虛擬機 (JVM) 內存和垃圾回收狀況。這個工具對於監測和診斷 JVM 性能表現十分有用，特別是在優化內存使用和理解垃圾回收行為方面。以下是 jstat 的使用方式和一些常見的選項。

## 基本用法

jstat 的基本語法如下：

```
jstat [option] <pid> [interval] [count]
```

- option：選擇要查看的統計信息類型。
- pid：要監控的 Java 進程的進程 ID (可以用 jps 查找)。
- interval (可選)：更新統計信息的間隔時間 (以毫秒為單位)。
- count (可選)：輸出更新的次數。

## 常見選項

- jstat -gc <pid>：顯示 JVM 垃圾回收相關的統計信息。

```
jstat -gc <pid> 1000 5
```

這會每隔 1000 毫秒 (1 秒) 更新一次垃圾回收的數據，共更新 5 次。輸出內容包含各個垃圾回收區的大小、使用情況，以及 YGC (年輕代垃圾回收) 和 FGC (老年代垃圾回收) 的次數等。

輸出示例：

S0C	S1C	S0U	S1U	EC	EU	OC	OU	MC	MU	CCSC	CCSU	YGC	YGCT	FGC	FGCT
GCT															
512.0	512.0	0.0	0.0	4352.0	1234.0	10240.0	5678.0	2560.0	1900.5	224.0	150.0	5	0.345	1	0.654
1.0															

各列解釋：

- S0C, S1C：第 0、1 Survivor 空間的容量 (KB)。
  - S0U, S1U：第 0、1 Survivor 空間的使用量 (KB)。
  - EC, EU：Eden 空間的容量和使用量。
  - OC, OU：Old 空間的容量和使用量。
  - MC, MU：元數據空間 (Metaspace) 的容量和使用量。
  - CCSC, CCSU：壓縮類空間的容量和使用量。
  - YGC, YGCT：年輕代 GC 的次數及時間。
  - FGC, FGCT：老年代 GC 的次數及時間。
  - GCT：總 GC 時間。
- jstat -gcutil <pid>：顯示垃圾回收區的使用百分比。

```
jstat -gcutil <pid> 1000 5
```

這會顯示各個區域使用率的百分比，方便快速查看內存使用情況。

- jstat -class <pid>：顯示加載類的統計數據。

```
jstat -class <pid> 1000 5
```

這會顯示 JVM 中類的加載數量、總字節數、卸載數量等。

- jstat -compiler <pid>：顯示 JIT 編譯器的統計信息。

```
jstat -compiler <pid> 1000 5
```

這會顯示 JIT 編譯的次數和時間，有助於了解編譯行為的情況。

## 實用範例

- 查看特定進程的垃圾回收行為：

```
jstat -gc 12345 2000
```

每隔 2 秒 (2000 毫秒) 更新一次數據，直到手動中斷。

- 查看特定進程的內存區域使用率百分比：

```
jstat -gcutil 12345 500 10
```

每隔 0.5 秒更新一次，共更新 10 次。

# 小結

`jstat` 是一個高效且簡單的工具，可以讓開發者和運維人員更深入地了解 JVM 的內存使用和垃圾回收情況。在診斷性能瓶頸和內存相關問題時，`jstat` 是不可或缺的工具。