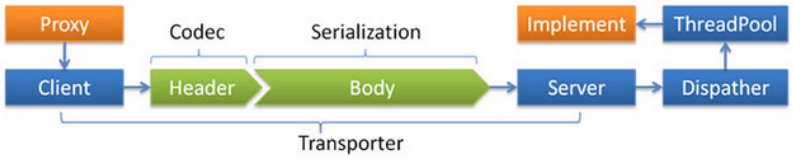
Dubbo 线程模型（结合 Linux 线程数限制配置的实战经验分享）

Dubbo 官方文档：

用户指南 >> 示例 >> 线程模型

线程模型

(+) (#)



事件处理线程说明

- 如果事件处理的逻辑能迅速完成，并且不会发起新的IO请求，比如只是在内存中记个标识，则直接在IO线程上处理更快，因为减少了线程池调度。
- 但如果事件处理逻辑较慢，或者需要发起新的IO请求，比如需要查询数据库，则必须派发到线程池，否则IO线程阻塞，将导致不能接收其它请求。
- 如果用IO线程处理事件，又在事件处理过程中发起新的IO请求，比如在连接事件中发起登录请求，会报"可能引发死锁"异常，但不会真死锁。

- Dispatcher
  - all 所有消息都派发到线程池，包括请求，响应，连接事件，断开事件，心跳等。
  - direct 所有消息都不派发到线程池，全部在IO线程上直接执行。
  - message 只有请求响应消息派发到线程池，其它连接断开事件，心跳等消息，直接在IO线程上执行。
  - execution 只请求消息派发到线程池，不含响应，响应和其它连接断开事件，心跳等消息，直接在IO线程上执行。
  - connection 在IO线程上，将连接断开事件放入队列，有序逐个执行，其它消息派发到线程池。
- ThreadPool
  - fixed 固定大小线程池，启动时建立线程，不关闭，一直持有。(缺省)
  - cached 缓存线程池，空闲一分钟自动删除，需要时重建。
  - limited 可伸缩线程池，但池中的线程数只会增长不会收缩。(为避免收缩时突然来了大流量引起的性能问题)。

配置如：

```
<dubbo:protocol name="dubbo" dispatcher="all" threadpool="fixed" threads="100" />
```

配置标签：

<dubbo:provider/>

<dubbo:protocol/>

<dubbo:protocol/>

(+) (#)

服务提供者协议配置：
配置类：com.alibaba.dubbo.config.ProtocolConfig
说明：如果需要支持多协议，可以声明多个<dubbo:protocol>标签，并在<dubbo:service>中通过protocol属性指定使用的协议。

标签	属性	对应URL参数	类型	是否必填	缺省值	作用	描述	兼容性
<dubbo:protocol>	threadpool	threadpool	string	可选	fixed	性能调优	线程池类型，可选：fixed/cached	2.0.5以上版本
<dubbo:protocol>	threads	threads	int	可选	100	性能调优	服务线程池大小(固定大小)	2.0.5以上版本
<dubbo:protocol>	iothreads	threads	int	可选	cpu个数+1	性能调优	io线程池大小(固定大小)	2.0.5以上版本
<dubbo:protocol>	accepts	accepts	int	可选	0	性能调优	服务提供方最大可接受连接数	2.0.5以上版本
<dubbo:protocol>	dispatcher	dispatcher	string	可选	dubbo协议缺省为all	性能调优	协议的消息派发方式，用于指定线程模型，比如：dubbo协议的all, direct, message, execution, connection等	2.1.0以上版本
<dubbo:protocol>	queues	queues	int	可选	0	性能调优	线程池队列大小，当线程池满时，排队等待执行的队列大小，建议不要设置，当线程池时应立即失败，重试其它服务提供机器，而不是排队，除非有特殊需求。	2.0.5以上版本

实战经验分享（属用性能调优）：

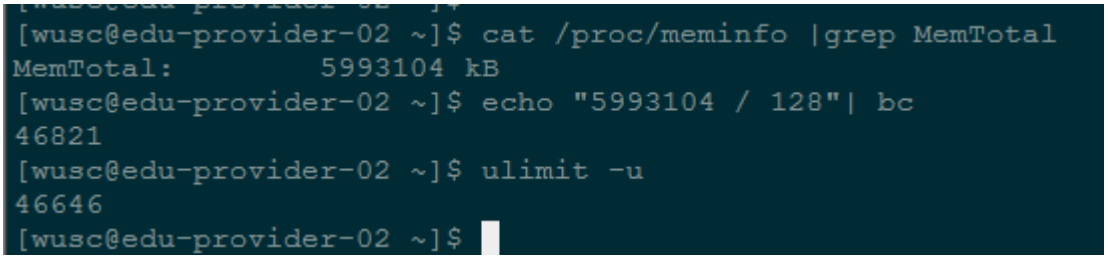
Linux 用户线程数限制导致的 java.lang.OutOfMemoryError: unable to create new native thread 异常

```
# vi /etc/security/limits.d/90-nproc.conf
# Default limit for number of user's processes to prevent
# accidental fork bombs.
# See rhbz #432903 for reasoning.
root          soft       nproc        unlimited
*             soft       nproc        20480
```

调整时要注意：

1、 尽量不要使用 root 用户来部署应用程序，避免资源耗尽后无法登录操作系统。

2、 普通用户的线程数限制值要看可用物理内存容量来配置



计算方式：

default_nproc = total_memory/128K;

```
$ cat /proc/meminfo |grep MemTotal
$ echo "5993104 / 128"| bc
$ ulimit -u
```

ulimit -a # 显示目前资源限制的设定

ulimit -u # 用户最多可开启的程序数目

重启，使之生效：# reboot