CHINA LINUX KERNEL
中国Linux内核开发者大会

vivo

使用MPTCP实现网络加速

报告人：唐葛亮

日　期：2025.11.1

目录
CONTENTS

01

个人简介

个人简介

唐葛亮

麒麟软件研发技术专家

Linux内核MPTCP
Maintainer

openEuler技术委员会委员
(2025-2026届)

Linux内核开发　MPTCP协议栈

Linux内核上游贡献

2015年起向Linux内核主线贡献补丁

900+

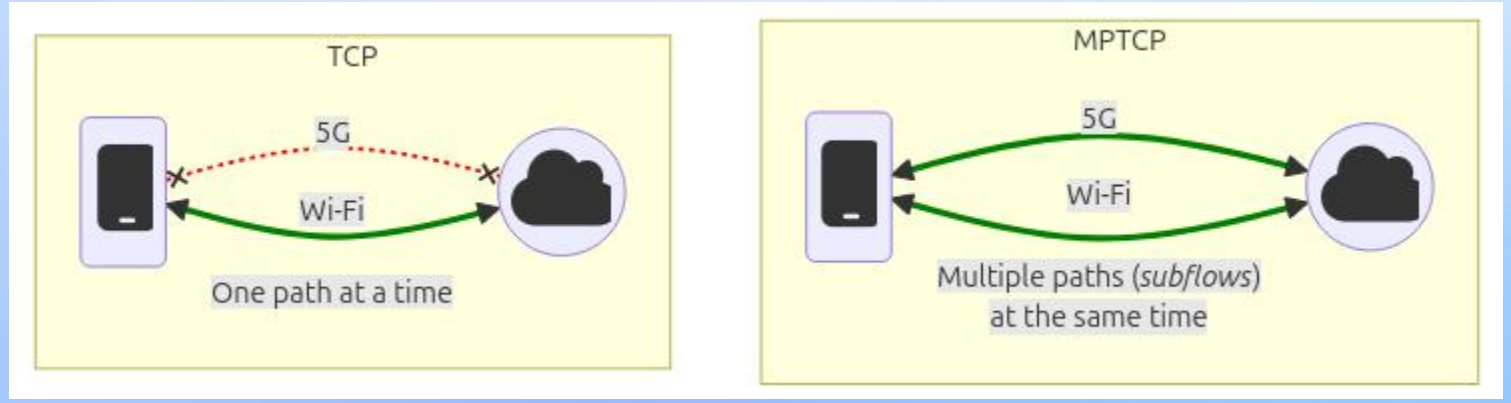
2019年起专注Linux内核MPTCP网络协议栈的开发，为该模块的核心开发者、主要贡献者、Reviewer及Maintainer

02

MPTCP 技术简介

MPTCP 技术简介——多路径TCP传输协议

- 多路径TCP或MPTCP是标准TCP的扩展，在RFC 8684（TCP Extensions for Multipath Operation with Multiple Addresses）中描述，通过TCP Option实现（Kind Number 30）。
- 从Linux 5.6开始MPTCP进入主线内核，源码目录 net/mptcp/，网站：www.mptcp.dev
- 它允许设备同时使用多个接口通过一条MPTCP连接来发送和接收TCP数据包，其可以聚合多个接口的带宽或优先选择延迟最低的接口



01 带宽扩展

通过多个网络路径并行传输，MPTCP能够将数据传输的带宽提升至原有**单条通道的n倍**，提高数据传输的速度和效率。

02 故障转移

当某一条子流出现故障导致数据无法传输时，MPTCP能够将数据转移到其他可用的路径上，确保数据传输的连续性和完整性。

03 链路选择

MPTCP提供了路径管理器，可以根据实际需求来配置决定多张网卡的连接方式，支持用户态和内核态的路径管理器。

MPTCP 技术简介——应用适配

C语言

```c
//需要把TCP socket创建的地方，第三个参数更换为IPPROTP_MPTCP。
// 创建MPTCP socket
#include <sys/socket.h>
int s = socket(AF_INET, SOCK_STREAM, IPPROTO_MPTCP)
if (s < 0) /* fallback to TCP */
    s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
```

Golang

```go
//SetMultipathTCP 在 GoLang 1.21 中支持:
import ("net" "context")
    // Client
    d := &net.Dialer{}
    d.SetMultipathTCP(true)
    c, err := d.Dial("tcp", *addr)
    // ServerLc := &ListenConfig{}
    lc.SetMultipathTCP(true)
    ln, err := lc.Listen(context.Background(), "tcp", *addr)
    defer ln.Close()
```

Python

```python
# socket.IPPROTO_MPTCP 在 CPython 3.10 中支持:
import socket
try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM, socket.IPPROTO_MPTCP)
except:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM, socket.IPPROTO_TCP)
```

MPTCP 技术简介—— 系统配置（路径管理器）

命令行配置

iproute2包中已经集成了ip mptcp 命令，通过该命令来手动完成路径的配置。

1）设置subflow 和 add_addr_acceted的数量限制，最大值为8

ip mptcp limits set add_addr_accepted 8

ip mptcp limits set subflows 8

2）配置本地网卡的连接属性

ip mptcp endpoint add [ip_addr] [dev name] [id] [signal/subflow]

（服务端用signal标志，客户端用subflow标志）

如下是在一台机器上配置两张网卡ens3/ens4的方式：

ip mptcp endpoint add 10.44.76.151 id 1 dev ens3 signal

ip mptcp endpoint add 10.44.76.152 id 2 dev ens4 signal

自动化配置

mptcpd软件包可以监控当前网络环境的变化，自动化配置MPTCP的路径管理器。目前，主流的Linux发行版均已完成mptcpd适配。

1）配置文件 /etc/mptcpd/mptcpd.conf，可以通过调整配置文件来自定义mptcpd行为（subflow/signal）

2）systemctl restart mptcp.service

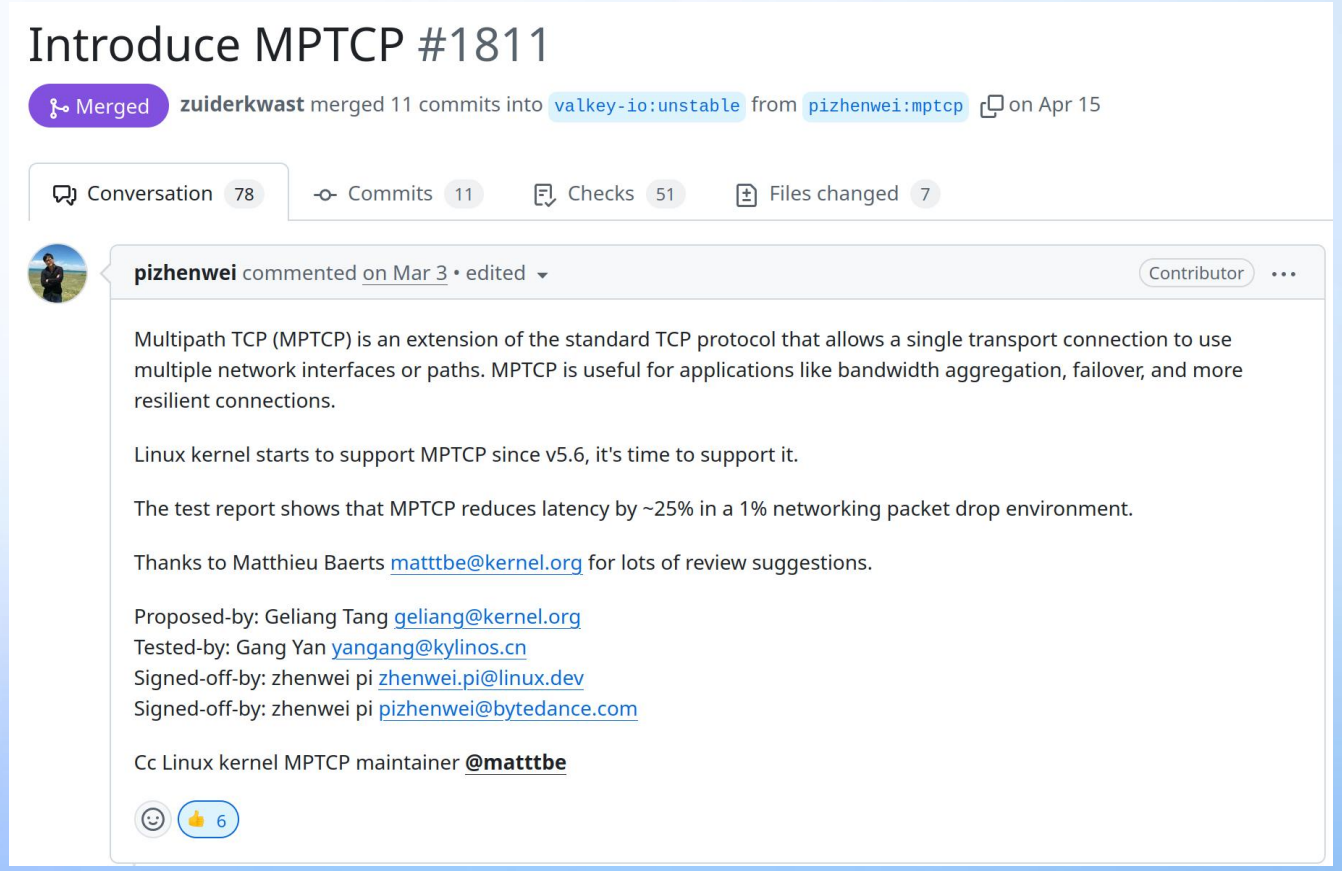
注：如配置mptcpd服务，则还需要将NetworkManager中的mptcp相关选项关闭，避免冲突。

03

MPTCP 应用实践

Valkey over MPTCP —— 场景背景

Valkey，作为Redis的重要开源分支，是一款被广泛使用的高性能KV数据库。在多网卡环境中使用Valkey over MPTCP，可以**提升数据库的并发量、降低延迟**。



Introduce MPTCP #1811

Merged **zuiderkwast** merged 11 commits into `valkey-io:unstable` from `pizhenwei:mptcp` on Apr 15

Conversation 78 Commits 11 Checks 51 Files changed 7

pizhenwei commented on Mar 3 • edited Contributor

Multipath TCP (MPTCP) is an extension of the standard TCP protocol that allows a single transport connection to use multiple network interfaces or paths. MPTCP is useful for applications like bandwidth aggregation, failover, and more resilient connections.

Linux kernel starts to support MPTCP since v5.6, it's time to support it.

The test report shows that MPTCP reduces latency by ~25% in a 1% networking packet drop environment.

Thanks to Matthieu Baerts matttbe@kernel.org for lots of review suggestions.

Proposed-by: Geliang Tang geliang@kernel.org
Tested-by: Gang Yan yangang@kylinos.cn
Signed-off-by: zhenwei pi zhenwei.pi@linux.dev
Signed-off-by: zhenwei pi pizhenwei@bytedance.com

Cc Linux kernel MPTCP maintainer **@matttbe**

6

Valkey over MPTCP——支持细节

代码展示

服务端

```
595 + static int _anetTcpServer(char *err, int port, char *bindaddr, int
    af, int backlog, int mptcp) {
596         int s = -1, rv;
597         char _port[6]; /* strlen("65535") */
598         struct addrinfo hints, *servinfo, *p;
ndaddr, int af, int backl
610             return ANET_ERR;
611         }
612         for (p = servinfo; p != NULL; p = p->ai_next) {
613 +           rv = anetTcpSetMptcp(err, p->ai_protocol, mptcp);
614 +           if (rv == ANET_ERR) goto error;
615 +
616 +           if ((s = socket(p->ai_family, p->ai_socktype, rv)) == -1)
    continue;
```

客户端

```
463     static int anetTcpGenericConnect(char *err, const char *addr, int
        port, const char *source_addr, int flags) {
464         int s = ANET_ERR, rv;
465         char portstr[6]; /* strlen("65535") + 1; */
addr, int port, const ch
482             * Make sure connection-intensive things like the benchmark
        tool
483             * will be able to close/open sockets a zillion of times.
484             */
485 +           int ai_protocol = anetTcpGetProtocol(flags &
        ANET_CONNECT_MULTIPATH);
486             int sockflags = ANET_SOCKET_CLOEXEC | ANET_SOCKET_REUSEADDR;
487             if (flags & ANET_CONNECT_NONBLOCK) sockflags |=
        ANET_SOCKET_NONBLOCK;
488 +           if ((s = anetCreateSocket(err, p->ai_family, p->ai_socktype,
        ai_protocol, sockflags)) == ANET_ERR) continue;
489             if (source_addr) {
490                 int bound = 0;
491                 /* Using getaddrinfo saves us from self-determining IPv4
        vs IPv6 */
```

```
static int anetCreateSocket(char *err, int domain, int type, int protocol,
    int s;

#ifdef SOCK_CLOEXEC
    if (flags & ANET_SOCKET_CLOEXEC) {
        type |= SOCK_CLOEXEC;
        flags &= ~ANET_SOCKET_CLOEXEC;
    }
#endif

#ifdef SOCK_NONBLOCK
    if (flags & ANET_SOCKET_NONBLOCK) {
        type |= SOCK_NONBLOCK;
        flags &= ~ANET_SOCKET_NONBLOCK;
    }
#endif

    if ((s = socket(domain, type, protocol)) == -1) {
        anetSetError(err, "creating socket: %s", strerror(errno));
        return ANET_ERR;
    }
```

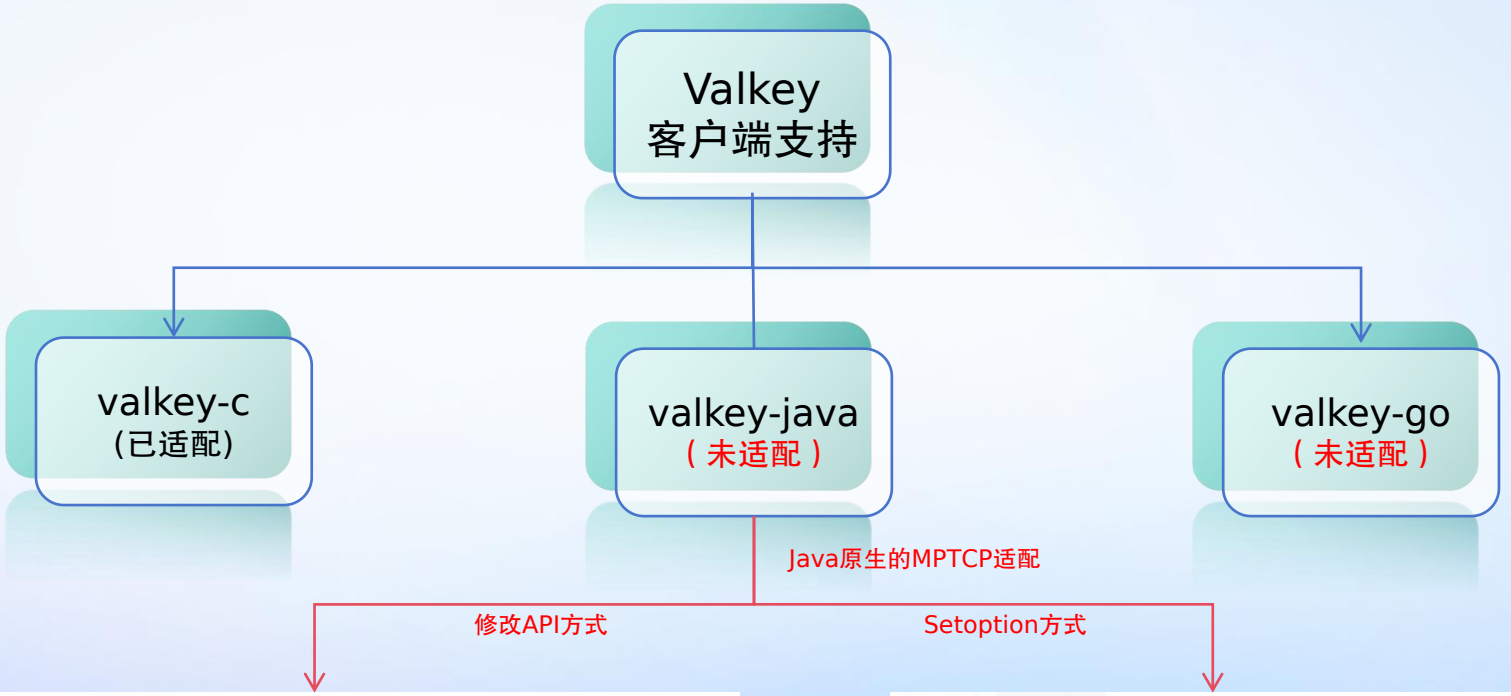
使用方法

自Valkey 9.0.0版本以后，上游版本原生集成MPTCP能力，启动时添加--mptcp参数即可启用。

eg:

```
valkey-cli --mptcp
valkey-benchmark --mptcp
valkey-server --mptcp yes
```

Valkey over MPTCP——TODO



Commit `0390b71`

geliangtang authored and **Geliang Tang** committed yesterday

```
Add MPTCP support to Java Socket/ServerSocket APIs

This patch integrates MPTCP support into the Java socket API by:

1. Adding new constructors in Socket and ServerSocket that accept an mptcp
   parameter to enable MPTCP explicitly
2. Extending NioSocketImpl to track the MPTCP state and use the appropriate
   native methods for socket creation
3. Propagating the MPTCP flag through the socket creation chain via new
   createImpl() factory methods
4. Ensuring backward compatibility by keeping existing constructors
   unchanged

The implementation allows Java applications to use MPTCP seamlessly while
maintaining full compatibility with existing TCP-based code.
```

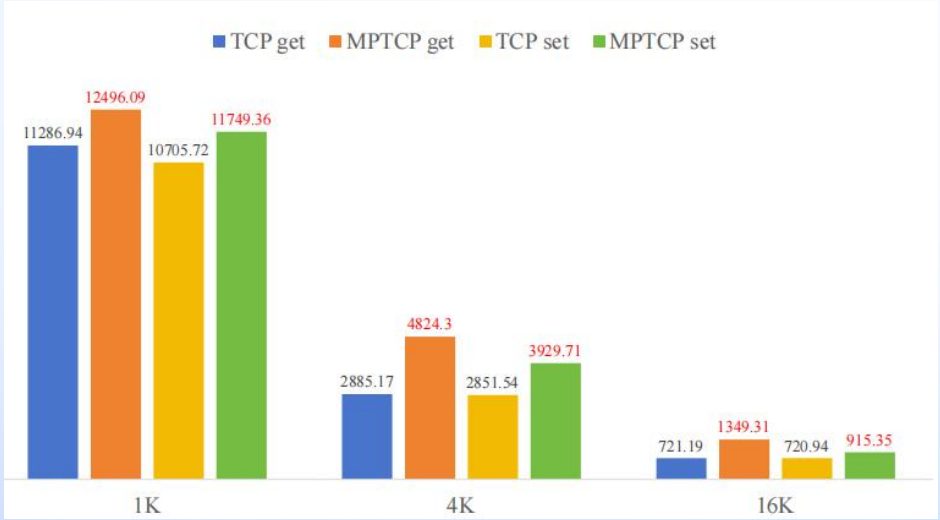
Commit `7c28d57`

geliangtang authored and **Geliang Tang** committed yesterday

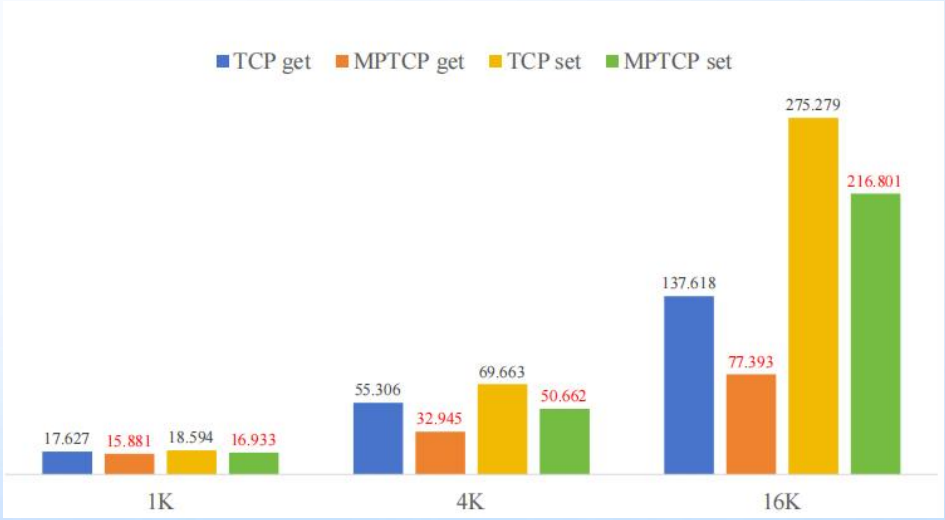
```
Add TCP_MPTCPIFY socket option for MPTCP support

Multipath TCP (MPTCP), as standardized in RFC 8684, is a major
evolution of the TCP protocol. It enables a transport connection to
use multiple network paths simultaneously for redundancy, resilience,
and bandwidth aggregation. Since its introduction in Linux kernel v5.6,
it has become a key technology for modern networking, particularly on
mobile devices with multiple interfaces (e.g., Wi-Fi and cellular).
For more details, see the project website: https://mptcp.dev.

This commit introduces the TCP_MPTCPIFY (the name was chosen to align
with 'mptcpize' tool in mptcpd and 'mptcpify' in BCC) socket option to
enable MPTCP support on Linux systems. The implementation includes:

- A new native method mptcpify0 in LinuxSocketOptions.c that creates
  an MPTCP socket and duplicates it to the original file descriptor,
  effectively converting the socket to use MPTCP. Relevant socket
  options are copied from the original socket to the new MPTCP socket.
- The TCP_MPTCPIFY option is exposed via ExtendedSocketOptions and is
  available only on Linux platforms.
- The feature is conditionally compiled using IPPROTO_MPTCP where
  supported.

This option must be set before calling bind() or connect() to take
effect.
```

Valkey over MPTCP——性能展示（2条100Mbit链路）

并发量（rpc）



延迟（latency）



基于官方性能测试工具 valkey-benchmark，对 Valkey over TCP 以及 MPTCP 进行测试。

测试涵盖了传输 1K、4K 以及 16K 数据量下的场景，重点测量了 get 和 set 操作的并发量 rpc以及延迟时间 latency。

NVMe over MPTCP —— 场景背景

NVMe over TCP是远程块存储协议，大文件传输时单路径TCP易达带宽上限，影响存储IO性能，属于内核层面使用MPTCP的典型案例。

nvmet-tcp: add mptcp support

This patch adds a new nvme target transport type NVMF_TRTYPE_MPTCP for MPTCP. And defines a new nvmet_fabrics_ops named nvmet_mptcp_ops, which is almost the same as nvmet_tcp_ops except .type.

Check if disc_addr.trtype is NVMF_TRTYPE_MPTCP in nvmet_tcp_add_port() to decide whether to pass IPPROTO_MPTCP to sock_create() to create a MPTCP socket instead of a TCP one.

This new nvmet_fabrics_ops can be switched in nvmet_tcp_done_recv_pdu() according to different protocol.

```
Co-Developed-by: Hui Zhu <zhuhui@kylinos.cn>
Signed-off-by: Hui Zhu <zhuhui@kylinos.cn>
Co-Developed-by: Gang Yan <yangang@kylinos.cn>
Signed-off-by: Gang Yan <yangang@kylinos.cn>
Co-Developed-by: zhenwei pi <zhenwei.pi@linux.dev>
Signed-off-by: zhenwei pi <zhenwei.pi@linux.dev>
Signed-off-by: Geliang Tang <tanggeliang@kylinos.cn>
```

2025-07-08	selftests: mptcp: add nvme over mptcp test
2025-07-08	nvme-tcp: add mptcp support
2025-07-08	mptcp: add mptcp_sock_set_reuseaddr
2025-07-08	nvmet-tcp: add mptcp support
2025-07-08	mptcp: add mptcp_sock_set_nodelay
2025-07-07	selftests: mptcp: add splice io mode
2025-07-07	mptcp: implement .splice_read
2025-07-07	mptcp: implement .read_sock
2025-07-07	mptcp: add eat_recv_skb helper

01

核心价值

大带宽场景下传输效率提升，IO上限突破单路径限制，多路径冗余保障存储IO不中断。

02

项目地址

git.kernel.org/pub/scm/linux/kernel/git/geliang/mptcp_net-next.git/log/?h=nvme-mptcp

NVMe over MPTCP——支持细节

代码展示（内核态MPTCP应用）

服务端

```
2035 + #ifdef CONFIG_MPTCP
2036 +     if (nport->disc_addr.trtype == NVMF_TRTYPE_MPTCP)
2037 +         proto = IPPROTO_MPTCP;
2038 + #endif
2039 +
2040         ret = inet_pton_with_scope(&init_net, af, nport->disc_addr.traddr,
2041                 nport->disc_addr.trsvcid, &port->addr);
2042         if (ret) {

2051             port->nport->inline_data_size = NVMET_TCP_DEF_INLINE_DATA_SIZE;
2052
2053         ret = sock_create(port->addr.ss_family, SOCK_STREAM,
2054 +                 proto, &port->sock);
```

客户端

```
diff --git a/drivers/nvme/host/tcp.c b/drivers/nvme/host/tcp.c
index f7e115cf131b3a..456ceb9d0bafb7 100644
--- a/drivers/nvme/host/tcp.c
+++ b/drivers/nvme/host/tcp.c
@@ -1764,6 +1764,7 @@ static int nvme_tcp_alloc_queue(struct nvme_ctrl *nctrl, int qid,
 {
         struct nvme_tcp_ctrl *ctrl = to_tcp_ctrl(nctrl);
         struct nvme_tcp_queue *queue = &ctrl->queues[qid];
+        int proto = IPPROTO_TCP;
         int ret, rcv_pdu_size;
         struct file *sock_file;

@@ -1780,9 +1781,14 @@ static int nvme_tcp_alloc_queue(struct nvme_ctrl *nctrl, int qid,
                 queue->cmnd_capsule_len = sizeof(struct nvme_command) +
                                                 NVME_TCP_ADMIN_CCSZ;

+#ifdef CONFIG_MPTCP
+        if (!strcmp(ctrl->ctrl.opts->transport, "mptcp"))
+                proto = IPPROTO_MPTCP;
+#endif
+
         ret = sock_create_kern(current->nsproxy->net_ns,
                                ctrl->addr.ss_family, SOCK_STREAM,
-                               IPPROTO_TCP, &queue->sock);
+                               proto, &queue->sock);
         if (ret) {
                 dev_err(nctrl->device,
                                "failed to create socket: %d\n", ret);
```

TODO

- **MPTCP TLS支持**

  目前MPTCP和TLS的底层架构有冲突，后续需要扩展支持该功能特性

- **MPTCP zerocopy支持**

  MPTCP协议的零拷贝接口实现和优化，提升性能

NVMe over MPTCP——FIO性能展示(2条1000Mbit链路)

TCP

```
$ sudo fio --name=global --direct=1 --norandommap --randrepeat=0 --ioengine=libaio --thread=1 --blocksize=4k --runtime=60 --time_
jobs=4 --iodepth=256 --group_reporting --size=100% --name=libaio_4_256_4k_randread --filename=/dev/nvme1n1

libaio_4_256_4k_randread: (g=0): rw=randread, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=256
...
fio-3.36
Starting 4 threads
Jobs: 4 (f=4): [r(4)][100.0%][r=109MiB/s][r=27.9k IOPS][eta 00m:00s]
libaio_4_256_4k_randread: (groupid=0, jobs=4): err= 0: pid=4090: Thu May 15 16:28:28 2025
  read: IOPS=28.0k, BW=109MiB/s (115MB/s)(6553MiB/60010msec)
    slat (usec): min=2, max=25408, avg=141.56, stdev=399.67
    clat (usec): min=811, max=81179, avg=36485.00, stdev=11613.30
     lat (usec): min=822, max=81186, avg=36626.56, stdev=11635.96
    clat percentiles (usec):
     |  1.00th=[ 7635],  5.00th=[15926], 10.00th=[20841], 20.00th=[26608],
     | 30.00th=[30802], 40.00th=[34341], 50.00th=[37487], 60.00th=[40109],
     | 70.00th=[43254], 80.00th=[46400], 90.00th=[50594], 95.00th=[54264],
     | 99.00th=[61080], 99.50th=[63701], 99.90th=[69731], 99.95th=[71828],
     | 99.99th=[76022]
   bw (  KiB/s): min=95016, max=133680, per=99.99%, avg=111814.99, stdev=1889.43, samples=476
   iops        : min=23754, max=33420, avg=27953.75, stdev=472.36, samples=476
  lat (usec)   : 1000=0.01%
  lat (msec)   : 2=0.08%, 4=0.22%, 10=1.44%, 20=7.17%, 50=79.77%
  lat (msec)   : 100=11.30%
  cpu          : usr=0.62%, sys=5.13%, ctx=339864, majf=0, minf=1028
  IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.1%
     issued rwts: total=1677589,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=256

Run status group 0 (all jobs):
   READ: bw=109MiB/s (115MB/s), 109MiB/s-109MiB/s (115MB/s-115MB/s), io=6553MiB (6871MB), run=60010-60010msec

Disk stats (read/write):
  nvme1n1: ios=1674281/0, sectors=13394248/0, merge=0/0, ticks=25237210/0, in_queue=25237210, util=100.00%
```

MPTCP（提升约94.78%）

```
$ sudo fio --name=global --direct=1 --norandommap --randrepeat=0 --ioengine=libaio --thread=1 --blocksize=4k --runtime=60 --time_
jobs=4 --iodepth=256 --group_reporting --size=100% --name=libaio_4_256_4k_randread --filename=/dev/nvme1n1

libaio_4_256_4k_randread: (g=0): rw=randread, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=256
...
fio-3.36
Starting 4 threads
Jobs: 4 (f=4): [r(4)][100.0%][r=214MiB/s][r=54.9k IOPS][eta 00m:00s]
libaio_4_256_4k_randread: (groupid=0, jobs=4): err= 0: pid=4048: Thu May 15 16:26:55 2025
  read: IOPS=54.7k, BW=214MiB/s (224MB/s)(12.5GiB/60050msec)
    slat (usec): min=2, max=29182, avg=71.56, stdev=306.20
    clat (usec): min=1045, max=246840, avg=18650.48, stdev=5890.22
     lat (usec): min=1091, max=246843, avg=18722.03, stdev=5898.96
    clat percentiles (usec):
     |  1.00th=[ 5473],  5.00th=[ 9372], 10.00th=[11469], 20.00th=[13960],
     | 30.00th=[15664], 40.00th=[17171], 50.00th=[18482], 60.00th=[19792],
     | 70.00th=[21365], 80.00th=[23200], 90.00th=[25822], 95.00th=[27919],
     | 99.00th=[33162], 99.50th=[36439], 99.90th=[53216], 99.95th=[58459],
     | 99.99th=[68682]
   bw (  KiB/s): min=194184, max=247112, per=100.00%, avg=218849.00, stdev=2535.25, samples=479
   iops        : min=48546, max=61778, avg=54712.25, stdev=633.81, samples=479
  lat (msec)   : 2=0.04%, 4=0.35%, 10=5.85%, 20=54.34%, 50=39.28%
  lat (msec)   : 100=0.14%, 250=0.01%
  cpu          : usr=1.18%, sys=8.18%, ctx=490869, majf=0, minf=1028
  IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=0.1%, >=64=100.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.1%
     issued rwts: total=3283542,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=256

Run status group 0 (all jobs):
   READ: bw=214MiB/s (224MB/s), 214MiB/s-214MiB/s (224MB/s-224MB/s), io=12.5GiB (13.4GB), run=60050-60050msec

Disk stats (read/write):
  nvme1n1: ios=3277041/0, sectors=26218232/0, merge=0/0, ticks=28443782/0, in_queue=28443782, util=100.00%
```

iperf3的 MPTCP支持

iperf3——快速验证

iperf3是开源网络性能测试工具，支持带宽、延迟、丢包率等指标测试。

3.19+版本支持MPTCP环境测速，用于MPTCP网络环境测试，**快速直观验证环境部署是否正确**。

eg:

iperf3 -s --mptcp

iperf3 -c <ip> -m

iperf-3.19 2025-05-16

iperf-3.19 2025-05-16

- Notable user-visible changes
  - iperf3 now supports the use of Multi-Path TCP (MPTCPv1) on Linux with the use of the `-m` or `--mptcp` flag. (PR #1661)
  - iperf3 now supports a `--cntl-ka` option to enable TCP keepalives on the control connection. (#812, #835, PR #1423)

add MPTCPv1 support #1661

Merged **bmah888** merged 1 commit into `esnet:master` from `geliangtang:master` on Jan 18

Conversation 5 | Commits 1 | Checks 3 | Files changed 8

geliangtang commented on Mar 12, 2024 — Contributor

The Multipath TCP (MPTCP) protocol (v1 / RFC 8684) has been added in the upstream Linux kernel since v5.6.

MPTCP is strongly tied to TCP, and the kernel APIs are almost the same. The only required dependency is the 'IPPROTO_MPTCP' protocol number definition, which should be provided by the netinet/in.h header if it is recent enough.

This patch adds a new flag '-m' or '--mptcp' to support MPTCPv1. It can be used like this:

```
iperf3 -m -s
iperf3 -m -c 127.0.0.1
```

There is no need to check for IPPROTO_MPTCP support in configure.ac at build time, it is at runtime we will see if the kernel being use supports or not MPTCP.

If IPPROTO_MPTCP is not supported by the kernel being tested, it is normal to fail because the feature is not available and the user explicitly asked to use MPTCP.

Closes: #1659
Co-developed-by: Paolo Abeni pabeni@redhat.com

BCC工具mptcpify

BCC——快速部署

基于BPF程序拦截TCP连接建立过程，动态替换传输层为MPTCP，实现无感升级与灵活可控。

0.35.0+版本后添加了mptcpify工具，可以拦截TCP套接字创建过程，动态替换传输层为MPTCP，**实现无感替换与灵活可控。**

eg:

mptcpify -t redis,iperf3

目前已验证rsync、iperf3、curl、redis、valkey可以通过该方式快速完成MPTCP的适配。

Update debian changelog for release v0.35.

```
  * Support for kernel up to 6.14
  * New bcc tools: mptcp: enable mptcp for tcp traffic
  * tools/biosnoop: Fix biosnoop pattern option
  * Allow cmake run out of the source tree
  * Fix for test bpf_stack_id when running on custom image in yocto
  * libbpf-tools/map_helpers: Add bpf_map_lookup_and_delete_batch t
  * libbpf-tools/biotop: Use dump_hash for map processing
  * uninstall: use execute_process() instead of exec_program()
  * clang: Fix pointer dereference on big-endian machines
  * ci: Upgrade to ubuntu 24.04
  * doc update, other bug fixes and example improvement.
```

```
+++ b/net/socket.c
@@ -1657,12 +1657,36 @@ struct file *__sys_socket_file(int family, int type, int pro
        return sock_alloc_file(sock, flags, NULL);
 }

+/*     A hook for bpf progs to attach to and update socket protocol.
+ *
+ *     A static noinline declaration here could cause the compiler to
+ *     optimize away the function. A global noinline declaration will
+ *     keep the definition, but may optimize away the callsite.
+ *     Therefore, __weak is needed to ensure that the call is still
+ *     emitted, by telling the compiler that we don't know what the
+ *     function might eventually be.
+ *
+ *     __diag_* below are needed to dismiss the missing prototype warning.
+ */
+
+__diag_push();
+__diag_ignore_all("-Wmissing-prototypes",
+                  "A fmod_ret entry point for BPF programs");
+
+__weak noinline int update_socket_protocol(int family, int type, int protocol)
+{
+       return protocol;
+}
+
+__diag_pop();
+
 int __sys_socket(int family, int type, int protocol)
 {
        struct socket *sock;
        int flags;

-       sock = __sys_socket_create(family, type, protocol);
+       sock = __sys_socket_create(family, type,
+                                  update_socket_protocol(family, type, protocol));
        if (IS_ERR(sock))
                return PTR_ERR(sock);
```

2023-08-16	Merge branch 'bpf: Force to MPTCP'
2023-08-16	selftests/bpf: Add mptcpify test
2023-08-16	selftests/bpf: Fix error checks of mptcp open_and_load
2023-08-16	selftests/bpf: Add two mptcp netns helpers
2023-08-16	bpf: Add update_socket_protocol hook

Python-BCC: support fmod_ret attaching m
add a useful tool for mptcp #5274

Merged **chenhengqi** merged 3 commits into `iovisor:master` from `Dwyane-Yan:fmod_ret`

Conversation 30 · Commits 3 · Checks 12 · Files changed 6

Dwyane-Yan commented on Apr 9 • edited — Contributor

Multipath TCP (MPTCP) serves as an enhancement to the conventional TCP protocol, enabling a single transport-layer connection to leverage multiple network interfaces. This capability makes MPTCP advantageous for applications requiring bandwidth consolidation, seamless failover mechanisms, and more robust connectivity solutions.

Linux kernel starts to support MPTCP since v5.6, and it provides a fmod_ret interface 'update_socket_protocol' to force applications using MPTCP instead of TCP without modifyiing its code.

So these patches provide a tool named 'mptcpify' which can achieve this. Using python-BCC is a more easy way for future development, so it is so important to support 'fmod_ret' in python-BCC.

The first patch is suggested by Yonghong:

""
On Sun, 2024-08-25 at 21:05 -0700, Yonghong Song wrote:
...
> Gang Yan, could you explore to add fmod_ret support in bcc? It should
> be similar to kfunc/kretfunc support. I am happy to review your patches.
""

when I try to make a commit to kernel. (Link attached below)
https://patchwork.kernel.org/project/netdevbpf/patch/tencent_1E619C9E44C8C4B2B713A0D6DD45B92BF70A@qq.com/

@matttbe from the MPTCP kernel team had a look at the new tool.

04

生态支持

生态支持——应用生态

Name	Version	How to use
Angie	1.10.0	`listen <address|port> multipath`
Apache HTTP Server	2.5.1	`Listen 80 options=multipathtcp` or `ProxyPass multipathtcp=On`
Apache Traffic Server	9.2.4	`server_ports: <port> mptcp`
Caddy	v2.7.4	Enabled by default since v2.10.0 (Go 1.24), controllable with `GODEBUG=multipath=` since v2.7.4 (Go 1.21)
cURL	8.9.0	`--mptcp`
dae	v0.8.0	`global { mptcp: true }`
Envoy	1.21.0	`"enable_mptcp": "true"` (`listening` config)
freenginx	1.27.5	`listen <address|port> multipath`
GoLang	1.21	Apps written in Go can easily enable MPTCP support. Since 1.24, it is enabled on the server side by default
HAProxy	3.1.0	`mptcp{,4,6}@<address>`
Lighttpd 1.4	1.4.76	`server.feature-flags = ( "server.network-mptcp" => "enable" )`
QEmu	6.1	`<ip>:<port>,mptcp`
Shadowsocks libev	v3.3.6	`--mptcp`
Shadowsocks Rust	v1.16.0	`--mptcp`
Shadowsocks Go	v1.9.0	`servers.tcpListeners.multipath`, `clients.multipathTCP`
sing-box	v1.4.0	`"tcp_multi_path": true`
SystemD	v257	`SocketProtocol=mptcp` (`[Socket]` section)
Traefik	v2.10.5	Enabled by default since v2.11.26 / v3.4.2 / v3.5.0-rc1 (Go 1.24), controllable with `GODEBUG=multipath=` since v2.10.5 (Go 1.21)
v2ray-core	v5.17.0	`"mptcp": true`
Valkey	v8.2.0	`mptcp yes`, `repl-mptcp yes` and `--mptcp`

生态支持—— 系统生态

Name	Version	mptcpd	Comments
ArchLinux	✅	✅	Rolling release: MPTCP support is enabled since kernel v5.6. `mptcpd` is packaged in AUR.
Alpine Linux	3.20	✅	`mptcpd` is available since v3.22.
Amazon Linux	2023	❌	`mptcpd` / `mptcpize` can be used from containers.
Azure Linux	3.0	❌	`mptcpd` / `mptcpize` can be used from containers.
Debian	12	✅	
Fedora	36	✅	
Gentoo	✅	✅	Rolling release.
Home Assistant OS	12.2	❌	`mptcpd` / `mptcpize` can be used from containers.
Kylin OS	V11	✅	
NixOS	✅	✅	`mptcpd` has been accepted in Jan. 2025
NixOS	✅	✅	`mptcpd` has been accepted in Jan. 2025
Open MPTCP Router	v0.60	✅	MPTCP-specific, where everything is automatically configured.
openEuler	24.03	✅	`mptcpd` 0.13 will be included in 24.03 SP3.
OpenWrt	24.10	✅	`ip-full` package is required to use `ip mptcp` command.
Raspberry Pi OS	20231004	✅	
RHEL	9	✅	MPTCP is available as a tech-preview since v8.3.
Slackware	✅	✅	Version: to be completed. `mptcpd` is packaged in SlackBuilds
SUSE	✅	❌	Version: to be completed.
Ubuntu	22.04	✅	
Void	✅	❌	Rolling release: MPTCP support is enabled since kernel v5.10.

05

联系方式

联系方式

GitHub：
@geliangtang；

邮箱：
geliang@kernel.org；

MPTCP技术社区：
官网：https://www.mptcp.dev/
讨论：https://github.com/multipath-tcp/mptcp_net-next
邮件列表：mptcp@lists.linux.dev