AIX 7.1 版

遠端直接存取記憶體

請注意

使用本資訊及其支援的產品之前，請先閱讀第 13 頁的『注意事項』中的資訊。

此版本適用於 AIX 7.1 版，以及所有後續版次和修改內容，除非新版中另有指示。

目錄

關於本文件

本文件將下列資訊提供給有經驗的 C 程式設計師：在 AIX® 作業系統中透過 Internet Wide Area RDMA Protocol (iWARP) 或「RDMA 網路介面控制器 (RNIC)」光纖，使用 Open Fabrics Enterprise Distribution (OFED) 動詞進行程式設計的詳細資訊。

若要有效率地使用文件，您應該熟悉指令、系統呼叫、子常式、檔案格式和特殊檔案。

強調顯示

下列為本文件所使用的強調顯示慣例：

粗體	指定指令、子常式、關鍵字、檔案、結構、目錄及系統已預先定義其名稱的其他項目。亦指定圖形物件，如使用者選取的按鈕、標籤及圖示。
斜體	指定由使用者提供實際名稱或值的參數。
等寬字體	指定特定資料值的範例、類似您可能看到顯示文字的範例、類似您以程式設計者身份可能寫出程式碼部分的範例、系統的訊息、或您應實際鍵入的資訊。

AIX 中區分大小寫

AIX 作業系統中的所有項目皆區分大小寫，亦即區分大寫和小寫字母。例如，您可以使用 **ls** 指令來列出檔案。如果您鍵入 LS，則系統會回應找不到該指令。另外，**FILEA**、**FiLea** 及 **filea** 即使位於相同目錄中，仍是三個不同的檔名。為了避免執行非預期的動作，請務必使用正確的大小寫。

ISO 9000

本產品的開發和製造過程使用 ISO 9000 註冊的品質系統。

遠端直接存取記憶體

有經驗的 C 程式設計師可以尋找在 AIX 作業系統中利用「遠端直接存取記憶體 (RDMA)」動詞和 Open Fabrics Enterprise Distribution (OFED) 動詞進行程式設計的詳細資訊。

若要有效率地使用資訊，您必須熟悉指令、系統呼叫、子常式、檔案格式和特殊檔案。

Open Fabrics Enterprise Distribution (OFED)

瞭解如何在 AIX 作業系統中開始進行 Open Fabrics Enterprise Distribution (OFED) 動詞程式設計。OFED 動詞容許需要高產量和低延遲以使用「遠端直接存取記憶體 (RDMA)」特性的應用程式。

OFED 概念

Open Fabrics Enterprise Distribution (OFED) 動詞的動詞層通用於 InfiniBand、RDMA over Converged Ethernet (RoCE)、Internet Wide Area RDMA Protocol (iWARP)，以及衍生自 InfiniBand 架構的動詞。

硬體需求

AIX 作業系統支援 RDMA over Converged Ethernet (RoCE) 配接卡。在 AIX 中支援 RoCE RDMA 的硬體稱為 PCIe2 10 GbE RoCE 配接卡。

軟體需求

「AIX OFED 動詞」是根據 OpenFabrics Alliance 的 OFED 1.5 程式碼。在 AIX 作業系統上，支援 OFED 程式碼的 32 位元和 64 位元使用者應用程式。下列程式庫隨附於安裝的 RDMA：

- Librdmacm
- Libibverbs

動詞 API

AIX 應用程式可以決定動詞 API，而動詞是必須與特定目的地進行通訊的 Open Fabrics Enterprise Distribution (OFED) 動詞或 AIX InfiniBand (IB) 動詞。

虛擬碼中的下列範例會測試必要遠端位址上 **rdma_resolve_addr** 指令的結果，以判定可使用的 OFED 動詞。

此程式會傳回下列值：

- **0-** 如果可以使用 OFED 動詞來建立與目的地的通訊。
- **error-** 如果無法透過 OFED 支援的裝置來建立與目的地的通訊，但是可以使用 InfiniBand 架構來建立通訊。

```
/*下列 check_ofed_verbs_support 常式會執行以下動作：
/*- 呼叫 rdma_create_event_channel，以開啟通道事件         */
/*- 呼叫 rdma_create_id()，以取得 cm_id                              */
/*- 然後呼叫 rdma_resolve_addr()                                                  */
/*- 取得通訊事件                                                                 */
/*- 傳回事件狀態：                                                                    */
/*    0：正
常
            */
/*    錯誤：NOK 輸出裝置可能不是 RNIC 裝置                    */
/*- 呼叫 rdma_destroy_id()，以刪除建立的 cm_id                       */
/*- 呼叫 rdma_destroy_event_channel，以關閉通道事件         */

        int check_ofed_verbs_support (struct sockaddr *remoteaddr)
            {
                                struct rdma_event channel *cm_channel;
                                struct rdma_cm_id *cm_id;
                                int ret=0;
                                cm_channel = rdma_create_event_channel();
```

```
                                if (!cm_channel)  {
                                                fprintf(stderr,"rdma_create_event_channel error
\n");
                                                return -1;
                        }
                                                ret = rdma_create_id(cm_channel, &cm_id, NULL,
RDMA_PS_TCP);
                                                if (ret) {
                                                        fprintf(stderr,"rdma_create_id:
%d\n", ret);
                                                rdma_destroy_event_channel(cm_channel);
                                                return(ret);
                        }
                                                ret = rdma_resolve_addr(cm_id, NULL,
remoteaddr, RESOLVE_TIMEOUT_MS);
                                                if (ret) {

fprintf(stderr,"rdma_resolve_addr: %d\n", ret);
                                                        goto out;
                        }
                                                ret = rdma_get_cm_event(cm_channel, &event);
                                                if (ret) {
                                                        fprintf(stderr,"
rdma_get_cm_event() failed\n");
                                                        goto out;
                        }
                        ret = event->status;
                                                rdma_ack_cm_event(event);
                out:
                                                rdma_destroy_id(cm_id);
                                                rdma_destroy_event_channel(cm_channel);
                                                return(ret);
            }
```

Libibverbs 程式庫

Libibverbs 程式庫可讓使用者空間處理程序使用「遠端直接存取記憶體 (RDMA)」動詞。

Libibverbs 程式庫會在 InfiniBand 架構規格和 RDMA 通訊協定動詞規格中予以說明。

數個 /dev/rdma/uverbsN 字元裝置節點是用來處理 **Libibverbs** 程式庫與 ib_uverbs 核心層之間的通訊。每個 RDMA 網路介面控制器 (RNIC) 配接卡都有一個已向 Open Fabrics Enterprise Distribution (OFED) 核心登錄的裝置（例如，uverbs1 和 uverbs2 裝置）。若要在適當的裝置上執行，程式庫會寫入與動詞相對應的指令。

相關資訊

InfiniBand

RDMA 通訊協定動詞

Librdmacm 程式庫

librdmacm 程式庫提供通訊管理程式 (CM) 功能，以及在不同光纖（例如，InfiniBand (IB)、RDMA over Converged Ethernet (RoCE) 或 Internet Wide Area RDMA Protocol (iWARP)）上執行的一組通用「遠端直接存取記憶體 (RDMA)」CM 介面。

不論存在的配接卡或埠數目為何，使用者空間都是使用單一 /dev/rdma/rdma_cm 裝置節點來與核心進行通訊。

必須在任何 RDMA 裝置上執行的應用程式會使用 **librdmacm** 程式庫。

RDMA 網路介面控制器 (RNIC)

具有 Internet Wide Area RDMA Protocol (iWARP) 和 Verbs 函數的網路 I/O 配接卡或內嵌式控制器。

RDMA_CM 通訊管理程式

「遠端直接存取記憶體」通訊管理程式 (RDMA_CM) 是用來設定可靠的連線，以傳送資料。

通訊管理程式提供 RDMA 傳輸中性介面，以建立連線。API 是根據 Socket 而來，但已針對佇列配對 (QP) 型語意進行調整。通訊是透過特定的 RDMA 裝置，而且資料傳送為訊息型作業。

RDMA CM 使用 **librdmacm** 程式庫來提供通訊管理，以設定和切斷 RDMA API 的連線。使用 **libibverbs** 程式庫進行資料傳送，通訊管理程式就可以與動詞 API 一起運作。

使用 OFED 動詞所管理的資源

列出使用 OFED 動詞所管理的資源。

完成佇列 (CQ)：

含有「完成佇列 (CQ)」的先進先出 (FIFO) 佇列。CQ 是與佇列配對相關聯，用來接收完成通知和事件。

完成佇列項目 (CQE)：

CQ 中的項目，可說明已完成「工作要求 (WR)」的相關資訊（例如，狀態和大小）。

事件通道：

用來報告通訊事件。每一個事件通道都會對映至檔案描述子。您可以使用和操作相關聯的檔案描述子（如任何其他檔案描述子）來變更其行為。您可以讓檔案描述子執行下列其中一項動作：

- 不封鎖檔案描述子
- 輪詢檔案描述子
- 選取檔案描述子

記憶體區域 (MR)：

一組已登錄存取權的記憶體緩衝區。若要搭配使用記憶體緩衝區與網路配接卡，則必須登錄記憶體區域。

保護網域 (PD)：

讓用戶端可以在網域內關聯多個資源（例如，佇列配對和記憶體區域）。然後，用戶端就可以將在保護網域內傳送或接收資料的存取權，授與位於 RDMA 光纖上的其他網域。

佇列配對 (QP)：

佇列配對 (QP) 包含傳送和接收佇列。傳送佇列會傳送要求 RDMA 作業的出埠訊息。接收佇列會接收送入訊息或立即資料。

分散或收集元素 (SGE)：

整個或局部本端登錄記憶體區塊的指標的項目。此元素保留區塊的起始位址、大小，以及具有相關聯許可權的 lkey。

分散或收集陣列：

存在於工作要求 (WR) 中的分散或收集元素陣列。此陣列的運作是根據作業碼，而作業碼收集來自多個緩衝區的資料並以單一串流形式傳送它們，或取得單一串流並將資料分散到數個緩衝區。

工作佇列 (WQ)：

工作佇列包含「傳送佇列」或「接收佇列」。工作佇列是用來傳送或接收訊息。

工作佇列元素 (WQE)：

「工作佇列元素」是工作佇列中的元素。

工作要求 (WR)：

「工作要求」是使用者張貼至工作佇列的要求。

通訊作業

列出可用於 RDMA 裝置的通訊作業。

傳送及立即傳送作業

傳送作業會將資料傳送至遠端「佇列配對 (QP)」的接收佇列。

若要接收資料，接收端必須將資料張貼至接收緩衝區。傳送端無法控制遠端主機中的資料。

立即 4 位元組值是與資料緩衝區一起傳輸。此立即值是在接收通知的過程中向接收端呈現，而且它未包含在資料緩衝區中。

接收作業

接收作業是傳送作業的相對應作業。

系統會通知接收主機，已接收到具有行內立即值的資料緩衝區。接收應用程式會維護接收緩衝區，並張貼資訊。

RDMA 讀取作業

RDMA 讀取作業會從遠端主機中讀取記憶體區域。

您必須指定從中複製讀取資訊的遠端虛擬位址和本端記憶體位址。在您執行「遠端直接存取記憶體 (RDMA)」作業之前，遠端主機必須提供適當的許可權才能存取其記憶體。設定許可權之後，不需要對遠端主機進行任何通知，就可以執行 RDMA 讀取作業。

原子作業
可用於 AIX 作業系統的「遠端直接存取記憶體 (RDMA)」硬體，不支援原子作業。

***RDMA* 寫入或 *RDMA* 立即寫入作業**
RDMA 寫入作業類似於 RDMA 讀取作業，但資料是寫入至遠端主機。

不需要對遠端主機進行任何通知，就可以執行 RDMA 寫入作業。RDMA 立即寫入作業則會將立即值通知遠端主機。

傳輸模式
傳輸模式會建立佇列配對的連線。

下列是支援的傳輸模式

- 可靠連線 (RC)
  - 每一個佇列配對 (QP) 都與另一個 QP 相關聯
  - 某個 QP 的傳送佇列所傳輸的訊息，可以可靠地傳遞給另一個 QP 的接收佇列。
  - 依順序遞送封包。
  - RC 類似於 TCP 連線。
- 不可靠的資料包 (UD)
  - 在 QP 之間未形成實際連線。
  - UD 模式類似於 UDP 連線。

規劃 Open Fabrics Enterprise Distribution (OFED)

在 /etc/libibverbs.d/ 目錄中，必須要有系統上安裝的每個「遠端直接存取記憶體 (RDMA)」配接卡的配置檔。

配置檔可讓 **libibverbs** 程式庫使用 RDMA 裝置的驅動程式。例如，若要使用 **Mellanox ConnectX-2 RoCE** 配接卡，mx2.driver 檔案必須存在於 /etc/libibverbs.d/ 目錄中。mx2.driver 檔案必須包含下列程式碼：

```
# cat /etc/libibverbs.d/mx2.driver
driver mx2
```

若要使用任何其他目錄（/etc/libibverbs.d/ 目錄除外），請使用 IBV_CONFIG_DIR 環境變數。若要建立兩個節點之間的通訊，配接卡必須已配置 IPv4 或 IPv6 位址。

使用通訊管理程式 (RDMA_CM) 來建立連線

「遠端直接存取記憶體 (RDMA)」RDMA_CM 通訊管理程式提供含有 RDMA 應用程式設計介面 (API) 的連線設定和切斷的通訊管理。

RDMA_CM 通訊管理程式是與 **libibverbs** 程式庫所定義的動詞 API 一起運作。**libibverbs** 程式庫提供傳送和接收資料所需的介面。

用戶端作業
瞭解主動或用戶端通訊的基本作業的概觀。

一般連線流程如下：

rdma_create_event_channel
建立要接收事件的通道。

rdma_create_id
配置概念上與 Socket 類似的 rdma_cm_id ID。

rdma_resolve_addr
取得要連接遠端位址的本端「遠端直接存取記憶體 (RDMA)」裝置。

rdma_get_cm_event
等待 RDMA_CM_EVENT_ADDR_RESOLVED 事件。

rdma_ack_cm_event
確認接收的事件。

rdma_create_qp
配置通訊的佇列配對 (QP)。

rdma_resolve_route
決定遠端位址的路徑。

rdma_get_cm_event
等待 RDMA_CM_EVENT_ROUTE_RESOLVED 事件。

rdma_ack_cm_event
確認接收的事件。

rdma_connect
連接至遠端伺服器。

rdma_get_cm_event
等待 RDMA_CM_EVENT_ESTABLISHED 事件。

rdma_ack_cm_event
確認接收的事件。

ibv_post_send()
透過連線執行資料傳送。

rdma_disconnect
切斷連線。

rdma_get_cm_event
等待 RDMA_CM_EVENT_DISCONNECTED 事件。

rdma_ack_cm_event
確認事件。

rdma_destroy_qp
毀損 QP。

rdma_destroy_id
釋放 rdma_cm_id ID。

rdma_destroy_event_channel
釋放事件通道。

註：在此範例中，用戶端已起始切斷動作。不過，用戶端或伺服器作業可以起始切斷處理程序。

伺服器作業
瞭解可針對被動或伺服器通訊執行的基本作業。

一般連線流程如下：

rdma_create_event_channel
建立要接收事件的通道。

rdma_create_id
配置概念上與 Socket 類似的 rdma_cm_id ID。

rdma_bind_addr
設定事件所接聽的本端埠號。

rdma_listen
開始接聽連線要求。

rdma_get_cm_event
等待含有新 rdma_cm_id ID 的 RDMA_CM_EVENT_CONNECT_REQUEST 事件。

rdma_create_qp
在新的 rdma_cm_id ID 上，配置通訊的佇列配對 (QP)。

rdma_accept
接受連線要求。

rdma_ack_cm_event
確認事件。

rdma_get_cm_event
等待 RDMA_CM_EVENT_ESTABLISHED 事件。

rdma_ack_cm_event
確認事件。

ibv_post_send()
透過連線執行資料傳送。

rdma_get_cm_event
等待 RDMA_CM_EVENT_DISCONNECTED 事件。

rdma_ack_cm_event
確認事件。

rdma_disconnect
切斷連線。

rdma_destroy_qp
毀損 QP。

rdma_destroy_id
釋放已連接的 rdma_cm_id ID。

rdma_destroy_id
釋放接聽 rdma_cm_id ID。

rdma_destroy_event_channel
釋放事件通道。

RDMA_CM 通訊管理程式範例

瞭解在 LinuxConf.Europe 2007 會議期間向 Open Fabrics Enterprise Distribution (OFED) 社群呈現的範例。

相關資訊

呈現給 OFED 社群的範例

作用中用戶端的範例

用戶端為作用中的通訊作業的範例。

```
/*
 * 建置：
 * cc -o client client.c -lrdmacm -libverbs
 *
 * 用法：
 * client <servername> <val1> <val2>
 *
 * 連接至伺服器，透過「RDMA 寫入」傳送 val1 並透過「RDMA 傳送」傳送 val2，
 * 然後從伺服器接收回 val1+val2。
 */
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <arpa/inet.h>

#include <rdma/rdma cma.h>
enum   {
        RESOLVE TIMEOUT MS        = 5000,
};
struct pdata {
        uint64_t    buf va;
```

```
        uint32_t    buf rkey;
};

int main(int argc, char   *argv[ ])
{
   struct pdata                                    *server pdata;
   struct rdma_event channel        *cm_channel;
   struct rdma_cm_id                       *cm_id;
   struct rdma_cm_event                   *event;
   struct rdma_conn_param               conn_param = { };
   struct ibv_pd                             *pd;
   struct ibv_comp_channel            *comp_chan;
   struct ibv_cq                             *cq;
   struct ibv_cq                             *evt_cq;
   struct ibv_mr                             *mr;
   struct ibv_qp_init_attr            qp attr = { };
   struct ibv_sge                             sge;
   struct ibv_send_wr                       send_wr = { };
   struct ibv_send_wr                    *bad send wr;
   struct ibv_recv_wr                       recv wr = { };
   struct ibv_recv_wr                       *bad recv wr;
   struct ibv_wc                               wc;
   void                                                *cq context;
   struct addrinfo                            *res, *t;
   struct addrinfo                            hints = { .ai_family    = AF INET,

                                                .ai_socktype  = SOCK STREAM
                                        };
       int                                                 n;
    uint32_t                                        *buf;
       int                                                 err;

      /* 設定 RDMA CM 結構 */
                           cm_channel = rdma_create_event_channel();
                        if (!cm_channel)  return 1;
                        err = rdma_create_id(cm_channel, &cm_id, NULL, RDMA_PS_TCP);
                        if (err)
                                        return err;
                        n = getaddrinfo(argv[1], "20079", &hints, &res);
                        if (n < 0)
                                        return 1;

                        /* 解析伺服器位址及路徑 */
            for (t = res; t; t = t->ai next) {
                    err = rdma_resolve_addr(cm_id, NULL, t->ai_addr, RESOLVE_TIMEOUT_MS);
                                        if (!err)
                            break;
            }
                        if (err)
                                        return err;
                        err = rdma_get_cm_event(cm_channel, &event);
                        if (err)
                                        return err;
            if (event->event != RDMA_CM_EVENT_ADDR_RESOLVED)
                                        return 1;
                                    rdma_ack_cm_event(event);
                        err = rdma_resolve_route(cm_id, RESOLVE_TIMEOUT_MS);
                        if (err)
                                        return err;
                        err = rdma_get_cm_event(cm_channel, &event);
                        if (err)
                                        return err;
            if (event->event != RDMA_CM_EVENT_ROUTE_RESOLVED)
                                        return 1;
                                    rdma_ack_cm_event(event);

                        /* 建立動詞物件，因為我們知道要使用哪一個裝置 */
            pd = ibv_alloc_pd(cm_id->verbs);
                        if (!pd)
                                        return 1;
            comp chan = ibv_create_comp_channel(cm_id->verbs);
                        if (!comp_chan)
                                        return 1;
            cq = ibv_create_cq(cm_id->verbs, 2,NULL, comp_chan, 0);
                        if (!cq)
                                        return 1;
                     if (ibv_req_notify_cq(cq, 0))
                                        return 1;
                         buf = calloc(2, sizeof (uint32_t));
                        if (!buf)
                                        return 1;
```

```
                    mr = ibv_reg_mr(pd, buf,2 * sizeof(uint32_t), IBV_ACCESS_LOCAL_ WRITE);
                    if (!mr)
                                return 1;
                    qp_attr.cap.max        send_wr = 2;
                    qp_attr.cap.max        send_sge = 1;
                    qp_attr.cap.max        recv_wr = 1;
                    qp_attr.cap.max        recv_sge = 1;
                    qp_attr.send_cq        = cq;
                    qp_attr.recv_cq        = cq;
                    qp_attr.qp_type        = IBV_QPT_RC;
                    err = rdma_create_qp(cm_id, pd, &qp_attr);
                    if (err)
                                return err;
                    conn_param.initiator_depth = 1;
                    conn_param.retry_count     = 7;

                    /* 連接至伺服器 */
                    err = rdma_connect(cm_id, &conn_param);
                    if (err)
                                return err;
                    err = rdma_get_cm_event(cm_channel,&event);
                    if (err)
                                return err;
        if (event->event != RDMA_CM_EVENT_ESTABLISHED)
                                return 1;
        memcpy(&server_pdata, event->param.conn.private_data, sizeof server_pdata);
                                rdma_ack_cm_event(event);

                    /* Prepost 接收 */
                    sge.addr    =       (uintptr_t) buf;
                    sge.length                      = sizeof (uint32_t);
        sge.lkey    =       mr->lkey;
                    recv_wr.wr_id =     0;
                    recv_wr.sg_list =   &sge;
                    recv_wr.num_sge =   1;

        if (ibv_post_recv(cm_id->qp, &recv_wr, &bad_recv_wr))
                                return 1;

                    /* 寫入/傳送要新增的兩個整數 */
                    buf[0] = strtoul(argv[2], NULL, 0);
                    buf[1] = strtoul(argv[3], NULL, 0);
                    printf("%d + %d = ", buf[0], buf[1]);
                    buf[0] = htonl(buf[0]);
                    buf[1] = htonl(buf[1]);

                    sge.addr    =       (uintptr_t) buf;
                    sge.length                      = sizeof (uint32_t);
        sge.lkey                        =  mr->lkey;
                    send_wr.wr_id                   = 1;
                    sendwr.opcode                   = IBV_WR_RDMA_WRITE;
                    send _wr.sg_list                = &sge;
                    send_wr.num_sge                 = 1;
                    send_wr.wr.rdma.rkey            = ntohl(server_pdata.buf_rkey);
                    send_wr.wr.rdma.remote_addr     = ntohll(server_pdata.buf_va);

    if (ibv post send(cm id->qp, &send_wr,  &bad_send_wr))
                            return 1;
                    sge.addr                        = (uintptr_t) buf + sizeof (uint32_t);
                    sge.length                      = sizeof (uint32_t);
        sge.lkey                        = mr->lkey;
                    send_wr.wr_id                   = 2;
                    send_wr.opcode                  = IBV_WR_SEND;
                    send_wr.send_flags              = IBV_SEND_SIGNALED;
                    send_wr.sg_list                 =&sge;
                    send_wr.num_sge                 = 1;

    if (ibv_post_send(cm_id->qp, &send_wr,&bad_send_wr))
                            return 1;

                    /* 等待接收完成 */
                    while (1) {
     if (ibv_get_cq_event(comp_chan,&evt_cq, &cq_context))
                                return 1;
                    if (ibv_req_notify_cq(cq, 0))
                                return 1;
                    if (ibv_poll_cq(cq, 1, &wc) != 1)
                                return 1;
                    if (wc.status != IBV_WC_SUCCESS)
                                return 1;
                    if (wc.wr_id == 0) {
```

```
                                        printf("%d\n", ntohl(buf[0]));
                    return 0;
            }
    }
    return 0;
}
```

OFED 指令

瞭解 Open Fabrics Enterprise Distribution (OFED) 指令，包括語法陳述式、旗標說明和用法範例。

ibv_devices 指令

列出可從使用者空間使用的「遠端直接存取記憶體 (RDMA)」裝置。

ibv_devinfo 指令

列印可從使用者空間使用的 RDMA 網路介面控制器 (RNIC) 裝置的相關資訊。

語法

```
ibv_devinfo [-v] { [-d <dev>] [-i <port>] } | [-l]
```

旗標

項目	說明
-d *dev*	使用 *dev* RDMA 裝置。依預設，會使用找到的第一個裝置。
-i *port*	使用 RDMA 裝置的 *port* 埠。依預設，會使用所有埠。
-l	只列印 RDMA 裝置名稱。
-v	列印 RDMA 裝置的所有屬性。

ofedctrl 指令

載入和卸載 **ofed_core** 核心延伸。

語法

ofedctrl { [-k *KernextName*] -l|u|q } | [-c | -p *ParameterName=Value*] | -h

旗標

項目	說明
-c	如果編輯過檔案，則會重新載入配置檔。
-h	指定用法。
-k *KernextName*	指定核心延伸路徑。依預設，會使用 /usr/lib/drivers/ofed_core 路徑。
-l	載入核心延伸。
-p *ParameterName=Value*	直接在指令行上設定參數的值。 **註：**使用 -p 選項所設定的值不是持續值。-p 選項只會變更現行配置。它並不會更新配置檔。重新啟動系統之後，不會套用使用 -p 選項所進行的變更。
-q	指出是否載入核心延伸。
-u	卸載核心延伸。

rping 指令

使用 RDMA 乒乓測試，來測試 RDMA 通訊管理程式 (RDMA_CM) 的連線。

語法

```
rping -s [-v] [-V] [-d] [-P] [-a address] [-p port] [-C message_count] [-S message_size]
```

```
rping -c [-v] [-V] [-d] -a address [-p port] [-C message_count] [-S message_size]
```

說明

rping 指令會使用 **librdmacm** 程式庫，來建立兩個節點之間的可靠「遠端直接存取記憶體 (RDMA)」連線。**rping** 指令也會選擇性地執行節點之間的 RDMA 傳送，然後切斷連線。**rping** 指令會設定 RDMA_CM 連線，並執行 RDMA 乒乓測試。如需 **rping** 指令的相關資訊，請參閱 Open Source OpenFabrics Alliance OFED 1.4（網址為 http://www.openfabrics.org）。

旗標

項目	說明
-a *address*	指定要連結伺服器上連線的網址，以及指定要連接至用戶端的伺服器位址。
-c	以用戶端身分執行。
-C *message_count*	指定透過每一個連線傳送的訊息數目。預設值為 infinite。
-d	顯示除錯資訊。
-p	指定接聽伺服器的埠號。
-P	以持續模式執行伺服器。這容許多個 **rping** 用戶端連接至單一伺服器實例，而且除非刪除實例，否則都會執行伺服器。
-v	顯示連線測試資料。
-V	驗證連線測試資料。
-s	以伺服器身分執行。
-S *message_size*	指定傳送的每一個訊息的大小（以位元組為單位）。預設值為 100。

相關資訊

Openfabrics

使用者層次直接存取程式設計庫 (uDAPL)

「使用者直接存取程式設計庫 (uDAPL)」是一種在傳輸時執行的直接存取架構，而傳輸支援直接資料存取（例如，InfiniBand 和 RDMA 網路介面控制器 (RNIC)）。

DAT Collaborative 可指定 uDAPL 應用程式設計介面 (API)。來自 Open Fabrics 的 uDAPL 程式碼庫會移轉到 AIX 作業系統，而且是透過 GX++ HCA 和 4X DDR 擴充卡 (CFFh) InfiniBand 配接卡來支援。

相關概念

AIX 作業系統中支援的 uDAPL API
AIX 作業系統並未支援 DAT Collaborative 指定的所有「使用者直接存取程式設計庫 (uDAPL) API」。

uDAPL 的供應商特定屬性
瞭解 AIX 作業系統所支援的供應商特定屬性。支援 `delayed_ack_supported`、`vendor_extension`、`vendor_ext_version`、`debug_query` 及 `debug_modify` 屬性。

相關資訊

Datcollaborative

安裝 uDAPL

AIX 作業系統支援「使用者層次直接存取程式設計庫 (uDAPL)」2.0 版。

uDAPL 安裝映像檔以 **udapl.rte** 的形式隨附於擴充套件上，。 此映像檔附有 DAT 標頭檔（位於 /usr/include/dat 目錄中）。 安裝映像檔也附有 **libdat.a** 和 **libdapl.a** 程式庫。

應用程式包括 DAT 標頭檔，並與 /usr/include/dat 目錄中的 **libdat.a** DAT 程式庫鏈結。DAT 層決定適當的基礎傳輸特定程式庫。

AIX uDAPL 提供者會使用 dat.conf 檔案項目，向 DAT 暫存器登錄它自己。/etc/dat.conf 檔案是與預設項目一起出貨，而且此檔案具有項目格式的詳細資料。

uDAPL 程式庫支援 AIX 系統追蹤以進行事件除錯。uDAPL 系統追蹤會連接含有下列項目的 ID：5C3（適用於 DAPL 事件）、5C4（適用於 DAPL 錯誤事件）、5C7（適用於 DAT 事件）和 5C8（適用於 DAT 錯誤事件）。起始追蹤層次的修改方式是使用 DAT_TRACE_LEVEL 和 DAPL_TRACE_LEVEL 環境變數。這些環境變數接受 0 - 10 範圍內的值。追蹤的事件數目和資料量是隨著關鍵追蹤層次增加，如下所示：

```
TRC_LVL_ERROR  = 1
TRC_LVL_NORMAL = 3
TRC_LVL_DETAIL = 7
```

其他標準 AIX 服務功能特性（例如，AIX 錯誤日誌）是用來識別追蹤事件時發生的問題。基礎傳輸層的服務功能特性（例如，**ibstat** 指令和 InfiniBand 元件追蹤）也有助於分析問題。

DAT API 傳回可使用 /usr/include/dat/dat_error.h 檔案解碼的標準回覆碼。DAT Collaborative 的 uDAPL 規格中提供回覆碼的詳細說明。

AIX 作業系統中支援的 uDAPL API

AIX 作業系統並未支援 DAT Collaborative 指定的所有「使用者直接存取程式設計庫 (uDAPL) API」。

下列是一般業界 uDAPL 實作以及 AIX 作業系統所支援的 API。

下列是一般業界 uDAPL 實作以及 AIX 作業系統不支援的 API。

API	版本
dat_cr_handoff	// 在 DAT 2.0 中
dat_ep_create_with_srq	// 在 DAT 2.0 中
dat_ep_recv_query	// 在 DAT 2.0 中
dat_ep_set_watermark	// 在 DAT 2.0 中
dat_srq_create	// 在 DAT 2.0 中
dat_srq_post_recv	// 在 DAT 2.0 中
dat_srq_resize	// 在 DAT 2.0 中
dat_srq_set_lw	// 在 DAT 2.0 中
dat_srq_free	// 在 DAT 2.0 中
dat_srq_query	// 在 DAT 2.0 中

下列是 AIX 作業系統不支援的其他 API：

- dat_lmr_sync_rdma_read
- dat_lmr_sync_rdma_write
- dat_registry_add_provider

·dat_registry_add_provider

對於所有不支援的 API，AIX 作業系統遵循 DAT 規格中所述的特定機制來識別不支援的 API 清單。這些包括本身為零和特定 DAT_MODEL_NOT_SUPPORTED 回覆碼的 max_srq 屬性值。根據業界實作和 DAT 規格，可以針對函數傳回 DAT_NOT_IMPLEMENTED 代碼，而這是不支援的作業。

遠端記憶體區域 (RMR) 相關 API（例如，*dat_rmr_create*、*dat_rmr_bind*、*dat_rmr_free* 和 *dat_rmr_query*）支援會視基礎主機通道配接卡 (HCA) 功能而定，而且成功或失敗是取決於基礎 InfiniBand 架構。GX++ HCA 和 4X DDR 擴充卡 (CFFh) InfiniBand 配接卡目前不支援 RMR 作業。

相關概念

使用者層次直接存取程式設計庫 (uDAPL)

「使用者直接存取程式設計庫 (uDAPL)」是一種在傳輸時執行的直接存取架構，而傳輸支援直接資料存取（例如，InfiniBand 和 RDMA 網路介面控制器 (RNIC)）。

uDAPL 的供應商特定屬性

瞭解 AIX 作業系統所支援的供應商特定屬性。支援 delayed_ack_supported、vendor_extension、vendor_ext_version、debug_query 及 debug_modify 屬性。

相關資訊

uDAPL: User Direct Access Programming Library

uDAPL 的供應商特定屬性

瞭解 AIX 作業系統所支援的供應商特定屬性。支援 delayed_ack_supported、vendor_extension、vendor_ext_version、debug_query 及 debug_modify 屬性。

AIX 作業系統是 InfiniBand (IB) 架構的傳輸提供者，其包括供應商特定介面配接卡 (IA®) 和 delayed_ack_supported 屬性。 delayed_ack_supported 屬性的值是 **true** 或 **false**。 值為 **true** 時，與 IA 相關聯的端點具有可修改的提供者特定 delayed_ack 屬性。delayed_ack_supported 屬性為 **false** 時，無法變更提供者特定 delayed_ack 屬性的端點。提供者特定 delayed_ack 屬性的端點的預設值為 **false**。delayed_ack 屬性設為 **true**，方式是使用 dat_ep_modify 選項以啟用基礎 InfiniBand (IB) 主機通道配接卡 (HCA) 的延遲確認特性，而該主機通道配接卡是與端點相關聯的特定 InfiniBand 佇列配對的配接卡。此硬體特性不是由所有 HCA 實作，因此不適用於所有 IA。啟用此特性時，除非在伺服器的系統記憶體中偵測到資料傳送作業，否則會延遲 HCA 所傳送的確認。此處理程序會導致增加少量延遲。

對於除錯錯誤，uDAPL 程式庫支援 AIX 系統追蹤。起始追蹤層次的變更方式是使用 DAT_TRACE_LEVEL 和 DAPL_TRACE_LEVEL 環境變數。若要使用 API 來動態變更這些追蹤層次，請在 AIX 上使用動態追蹤層次支援。若要驗證程式庫是否支援動態追蹤層次，應用程式可以查詢供應商特定 IA vendor_extension 屬性。存在的 vendor_extension 屬性指出支援的動態追蹤層次。vendor_extension 屬性存在時，應用程式可以存取 dat_trclvl_query() 和 dat_trclvl_modify() 函數指標，方法是查詢 debug_query 和 debug_modify 供應商特定 IA 屬性。這些屬性的值指向相對應的函數。若要在未來可使用此 vendor_extension 介面，則必須使用 vendor_extension 供應商特定 IA 屬性。vendor_extension 屬性目前設為 1.0，而且它是唯一支援的版本。如果 vendor_extension 屬性不存在，則應用程式無法動態修改追蹤層次。

與 AIX 實作一起安裝的 uDAPL 範例程式碼，包括如何變更這些屬性的範例。

相關概念

AIX 作業系統中支援的 uDAPL API

AIX 作業系統並未支援 DAT Collaborative 指定的所有「使用者直接存取程式設計庫 (uDAPL) API」。

使用者層次直接存取程式設計庫 (uDAPL)

「使用者直接存取程式設計庫 (uDAPL)」是一種在傳輸時執行的直接存取架構，而傳輸支援直接資料存取（例如，InfiniBand 和 RDMA 網路介面控制器 (RNIC)）。

注意事項

本資訊係針對在美國所提供之產品與服務所開發。

在其他國家，IBM 不見得有提供本文件所提及之各項產品、服務或功能。請洽詢當地的 IBM 業務代表，以取得當地目前提供的產品和服務之相關資訊。本文件在提及 IBM 的產品、程式或服務時，不表示或暗示只能使用 IBM 的產品、程式或服務。只要未侵犯 IBM 之智慧財產權，任何功能相當之產品、程式或服務皆可取代 IBM 之產品、程式或服務。不過，任何非 IBM 之產品、程式或服務，使用者必須自行負責作業之評估和驗證責任。

本文件所說明之主題內容，IBM 可能擁有其專利或專利申請案。使用者不得享有本書內容之專利權。 您可以用書面方式來查詢特許權限，來函請寄到：

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

若要查詢有關雙位元組字元 (DBCS) 資訊的授權查詢事宜，請聯絡您國家的 IBM 智慧財產部門，或書面提出授權查詢，來函請寄到：

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

International Business Machines Corporation 只依「現況」提供本出版品，不提供任何明示或默示之保證， 其中包括但不限於不侵權、可商用性或特定目的之適用性的隱含保證。有些轄區在特定交易上，不允許排除明示或暗示的保證，因此，這項聲明不一定適合您。

本書中可能會有技術上或排版印刷上的訛誤。因此，IBM 會定期修訂； 並將修訂後的內容納入新版中。IBM 隨時會改進及/或變更本出版品所提及的產品及/或程式，不另行通知。

本資訊中任何對非 IBM 網站的敘述僅供參考，IBM 對該網站並不提供任何保證。該網站上的資料，並非本 IBM 產品所用資料的一部分，如因使用該網站而造成損害，其責任由 貴客戶自行負責。

IBM 得以各種 IBM 認為適當的方式使用或散佈由 貴客戶提供的任何資訊，而無需對 貴客戶負責。

如果本程式之獲授權人為了 (i) 在個別建立的程式和其他程式（包括本程式）之間交換資訊，以及 (ii) 相互使用所交換的資訊，因而需要相關的資訊，請洽詢：

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

上述資料之取得有其特殊要件，在某些情況下必須付費方得使用。

IBM 基於 IBM 客戶合約、IBM 國際程式授權合約或雙方之任何同等合約的條款，提供本文件所提及的授權程式與其所有適用的授權資料。

所引用的效能資料及客戶範例僅為說明用途。實際的效能結果可能會因為特定的配置與運作條件而有差異。

本書所提及之非 IBM 產品資訊，係一由產品的供應商，或其出版的聲明或其他 公開管道取得。IBM 並未測試過這些產品，也無法確認這些非 IBM 產品的執行效能、 相容性或任何對產品的其他主張是否完全無誤。如果您對非 IBM 產品的性能有任何的疑問，請逕向該產品的供應商查詢。

關於 IBM 未來方針或意向的之聲明，僅代表 IBM 的目標與目的，隨時可能變動或撤消，不另行通知。

本出版品中所顯示的所有 IBM 價格皆為 IBM 的現行建議零售價，隨時可能變更，恕不另行通知。經銷商的價格可能與此不同。

本資訊僅供規劃之用。在所述的產品上市之前，本文之資訊隨時可能更改。

本資訊中包含日常商業活動使用的資料與報告範例。為了盡可能完整地說明，範例中包括了個人、公司行號、品牌以及產品等的名稱。所有這些名稱全為虛構，任何與實際人員或商場企業類似之處，純屬巧合。

著作權：

本資訊含有原始語言之範例應用程式，用以說明各作業平台中之程式設計技術。貴客戶可以為了研發、使用、銷售或散布符合範例應用程式所適用的作業平台之應用程式介面的應用程式，以任何形式複製、修改及散布這些範例程式，不必向 IBM 付費。該等範例並未在一切情況下完整測試。因此，IBM 不保證或默示保證這些樣本程式之可靠性、服務性或功能。這些程式範例以「現狀」提供，且無任何保證。 IBM 對因使用這些程式範例而產生的任何損害概不負責。

這些程式範例的每一個拷貝或任何部分，或是任何的衍生著作，都必須包括下列的版權聲明：

©（貴公司名稱）（年）。

本程式碼之若干部分係衍生自 IBM 公司的範例程式。

© Copyright IBM Corp. _enter the year or years_.

隱私權條款考量

IBM® 軟體產品（包括軟體即服務解決方案，下面簡稱「軟體供應項目」）可能會使用 Cookie 或其他技術來收集產品使用情況資訊，以協助改良一般使用者體驗、修正與一般使用者的互動，或用於其他用途。在許多情況下，「軟體供應項目」並不會收集個人識別資訊。本公司的部分「軟體供應項目」可協助讓您收集個人識別資訊。 如果本「軟體供應項目」使用 Cookie 來收集個人識別資訊，以下將規定關於這類供應項目使用 Cookie 的特定資訊。

本「軟體供應項目」不會使用 Cookie 或其他技術來收集個人識別資訊。

如果本「軟體供應項目」所部署的配置向 貴客戶提供了通過 Cookie 及其他技術來收集一般使用者個人可識別資訊的能力， 貴客戶應該自行尋求關於此類資料收集所適用的任何法律建議，其中包括對於注意事項及同意的任何需求。

如需將各種技術（包括 Cookie）用於這些目的的相關資訊，請參閱《IBM 隱私權條款》(http://www.ibm.com/privacy) 和《IBM 線上隱私權聲明》(http://www.ibm.com/privacy/details) 以及 http://www.ibm.com/software/info/product-privacy 中標題為「Cookie、Web 訊號指標及其他技術」和「IBM 軟體產品及軟體即服務 (SaaS) 的隱私權聲明」的章節。

商標

IBM、IBM 標誌及 ibm.com 是 International Business Machines Corp. 在世界許多管轄區註冊的商標或註冊商標。其他產品及服務名稱可能是 IBM 或其他公司的商標。如需最新的 IBM 商標清單，請造訪位於 www.ibm.com/legal/copytrade.shtml 的 Copyright and trademark information 網頁。

INFINIBAND、InfiniBand Trade Association 及 INFINIBAND 設計標記是 INFINIBAND Trade Association 的商標及/或服務標記。

Linux 是 Linus Torvalds 在美國及（或）其他國家的註冊商標。