Dokumentácia ku knižnici pre medziprocesovú komunikáciu (IPC)

Kapitola 1. Používateľská príručka (API Reference)

NÁZOV

`libcom` — knižnica pre vysokovýkonnú medziprocesovú komunikáciu (IPC) založenú na POSIX zdieľanej pamäti s podporou Zero-Copy architektúry.

POUŽITIE

Pre použitie knižnice je potrebné vložiť hlavičkový súbor `com.h` a pri kompilácii linkovať program s príznakmi `-pthread` a `-lrt`.

POPIS FUNKCIÍ

Void com_initialize(intnr_proc, int*rank);
Inicializuje prostredie behu. Vytvorí `nr_proc` príbuzných procesov pomocou systémového volania fork(). Do premennej `rank` vráti unikátny logický identifikátor procesu (od 0 po nr_proc - 1). Volanie taktiež vykonáva automatické čistenie "osirelých" pamäťových segmentov z predchádzajúcich nekorektných ukončení.

void com_finalize(void);
Korektne ukončuje prácu knižnice. Uvoľňuje všetky lokálne a zdieľané zdroje, odstraňuje súbory z `/dev/shm` a ničí semafory. Hlavný proces (rank 0) automaticky čaká na ukončenie všetkých dcérskych procesov.

void* com_alloc_msg(size_t size);
Alokuje segment zdieľanej pamäte s veľkosťou `size` bajtov pre odoslanie správy. Vracia priamy ukazovateľ na alokovanú pamäť. Funkcia je blokujúca. Ak predchádzajúca správa procesu ešte nebola prečítaná príjemcom, proces bude pozastavený, kým sa pamäť neuvoľní.

void com_free_msg(void *msg);
Uvoľňuje prijatú správu. Signalizuje odosielateľovi, že dáta boli prečítané a pamäť môže byť znovu použitá.

void com_send(int rank, void *msg, size_t size);
Asynchrónne odosiela správu procesu s identifikátorom `target_rank`. Odovzdáva len metadáta; samotné kopírovanie dát sa nevykonáva. Funkcia vracia riadenie okamžite.

void com_mcast(void *msg, size_t size);
Odosiela správu hromadne (Multicast) všetkým ostatným procesom v systéme. Nevyžaduje žiadne duplikovanie dát v pamäti.

void com_recv(void **msg, size_t *size);
Čaká na prichádzajúcu správu (či už adresnú alebo multicast). V prípade úspechu zapíše do `*size` veľkosť dát a do `*msg` vráti priamy ukazovateľ na zdieľanú pamäť odosielateľa (Zero-Copy čítanie).

Kapitola 2. Vnútorná architektúra knižnice

Knižnica je navrhnutá s dôrazom na maximálnu priepustnosť, prevenciu vnútorných uviaznutí (Deadlocks) a minimalizáciu systémových volaní.

2.1. Koncept Zero-Copy

Na rozdiel od klasických IPC knižníc, ktoré využívajú prechodné buffery, táto implementácia úplne vylučuje použitie `memcpy` a `malloc` vo fáze prenosu dát. Pamäť je rozdelená na dve entity:

1 - Riadiaci blok (Control Block): Anonymný segment zdieľanej pamäte obsahujúci synchronizačné primitíva (POSIX semafory) a metadáta správ (adresy súborov, ranky príjemcov).

2 - Užitočné zaťaženie (Payload): Pomenované segmenty v `tmpfs` (virtuálny súborový systém `/dev/shm`), vytvárané cez `shm_open`. Funkcia com_recv() mapuje pamäť odosielateľa priamo do adresného priestoru príjemcu pomocou `mmap`. Čas prenosu správy má algoritmickú zložitosť O(1) a vôbec nezávisí od jej objemu.

2.2. Vzor "Schránka odosielateľa" (Outbox Pattern)

Pre zabránenie súbehu (Race Condition) pri hromadnom odosielaní správ je pole metadát priradené odosielateľom, nie príjemcom. Každý proces má právo zápisu len do svojho osobného slotu `messages[my_rank]`. Hlavnou výhodou tohto prístupu je mimoriadne efektívny Multicast (O(1)), pri ktorom nedochádza k žiadnemu duplikovaniu dát v pamäti, a procesy nemusia súperiť o globálny zámok (mutex) príjemcu.

2.3. Synchronizácia a bezpečnosť vlákien (Thread-Safety)

Na prebúdzanie procesov slúžia polia semaforov sem_recv() a sem_send(). Čakanie je implementované bez cyklov aktívneho čakania (Busy-waiting). Ochrana proti falošným prebudeniam (Spurious wakeups) a systémovým signálom (`EINTR`) je zabezpečená obalením systémových volaní do overovacích cyklov. * Knižnica je bezpečná pre vlákna (Thread-Safe). Použitie vnútorných mutexov a štruktúry `RecvTrack` umožňuje použitie viacerých vlákien (`pthreads`) v rámci jedného procesu pre paralelné prijímanie a odosielanie dát.

2.4. Vyhľadávanie Round-Robin (Ochrana proti vyhladovaniu)

Pri hľadaní prichádzajúcich správ používa príjemca algoritmus kruhového radu (Round-Robin), pričom hľadanie začína od nasledujúceho ranku po tom, ktorý bol prečítaný ako

posledný. To zaručuje absolútnu spravodlivosť (Fairness) a chráni procesy s vyššími rankami pred efektom vyhladovania (Starvation).

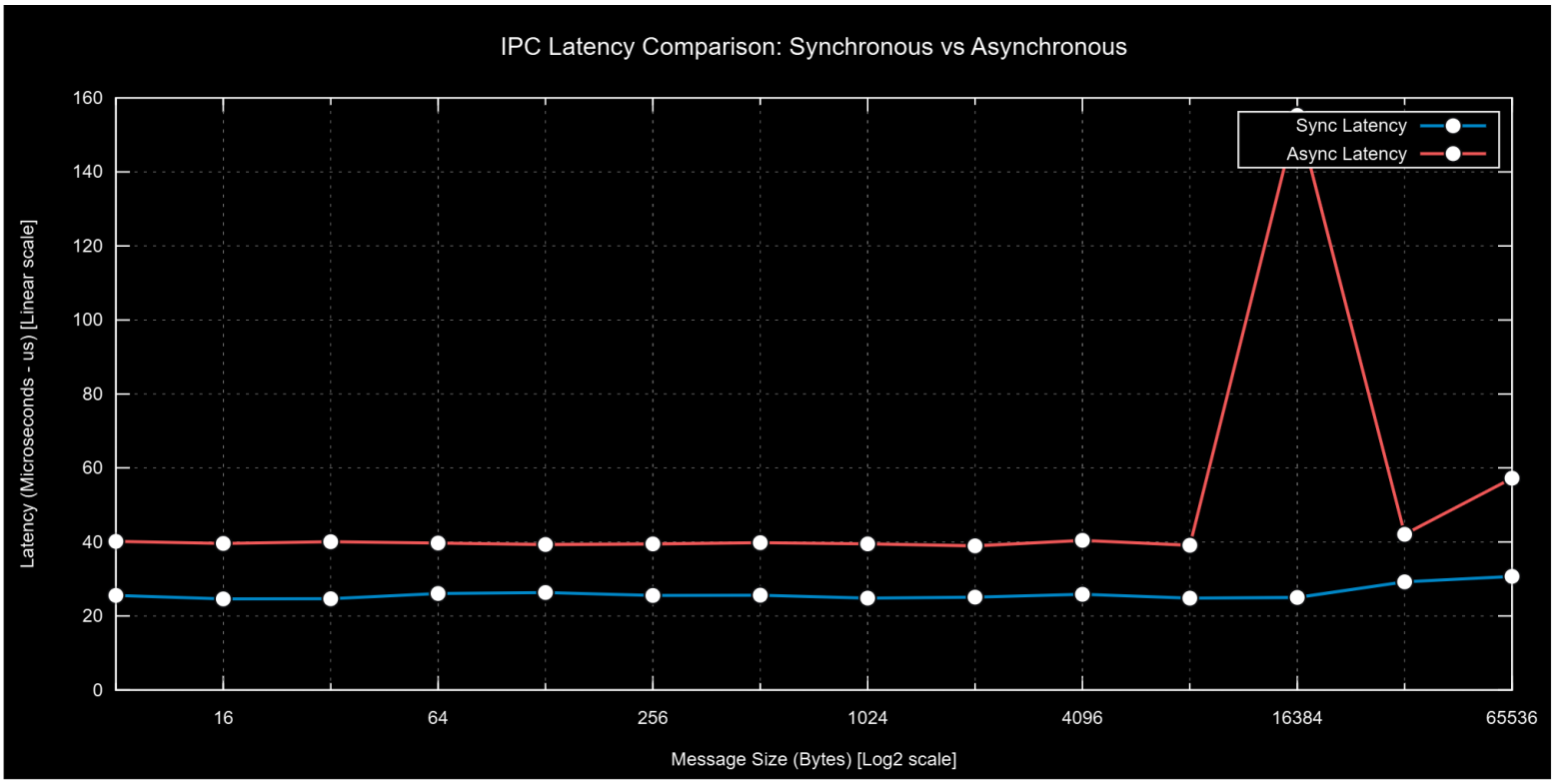
Kapitola 3. Popis benchmarkov a analýza výkonu Na vyhodnotenie výkonnosti knižnice boli vyvinuté dva benchmarky, ktoré simulujú rôzne topológie záťaže.

3.1. Synchrónny Benchmark: Multicast Ping-Pong (`benchmark.c`)

Tento test modeluje vzor "Master-Workers".

Topológia: Proces 0 hromadne odošle správu (Multicast) všetkým ostatným procesom. Každý pracovný proces prijme dáta a následne pošle adresnú odpoveď Masterovi.

Cieľ: Overenie správnosti fungovania počítadiel čítaní multicastu (`readers_left`) a meranie základnej latencie (Latency).



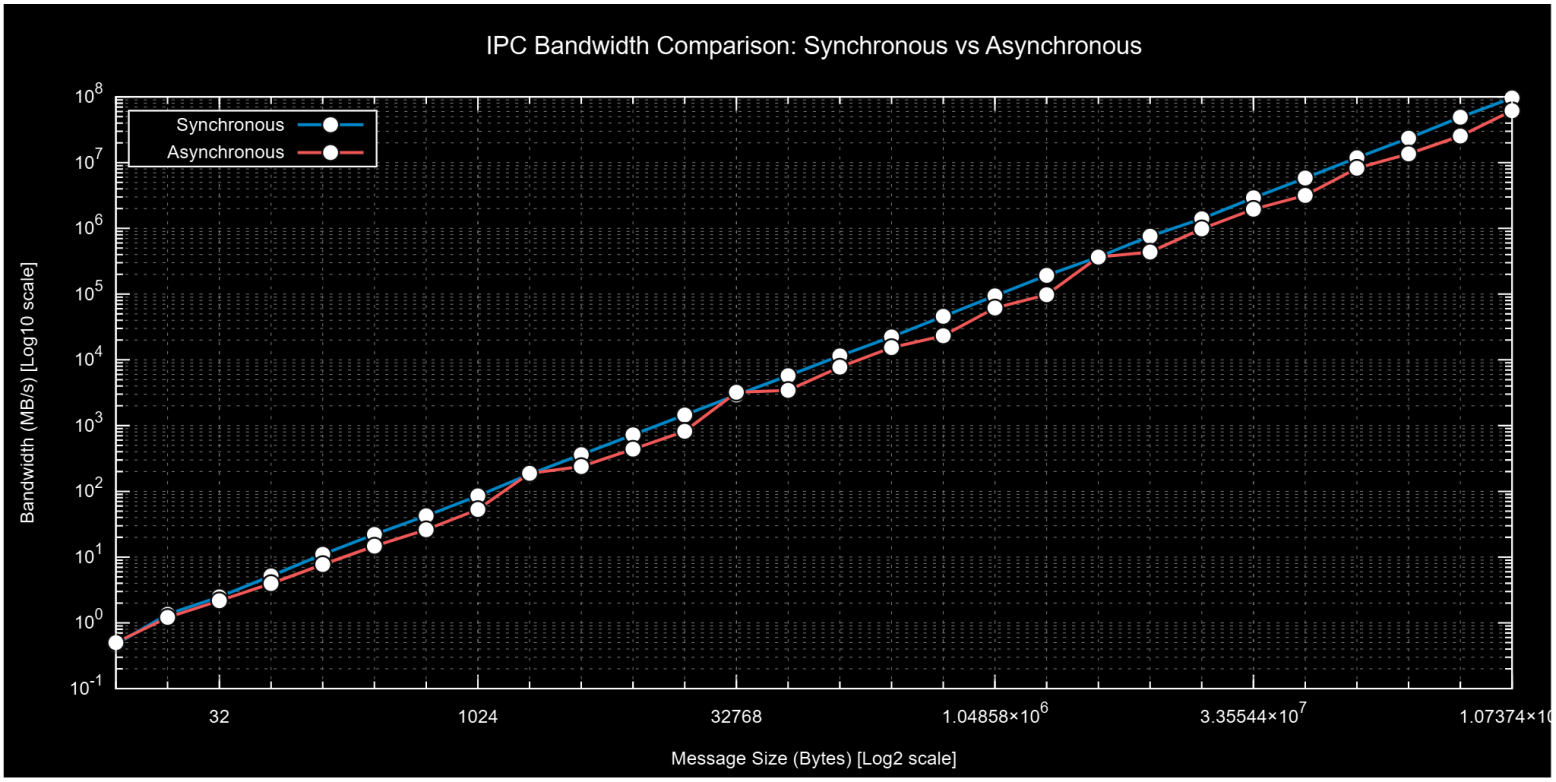
Na grafe latencie je vidieť, že pre správy menšieho objemu zostáva čas prenosu konštantný (horizontálna čiara). To demonštruje minimálnu réžiu (overhead) knižnice — čas sa spotrebúva výlučne na systémové volania prepínania kontextu plánovačom jadra Linux.

3.2. Asynchrónny Benchmark: Pairwise Duplex (`benchmark_async.c`)

Tento test modeluje prácu distribuovanej databázy s využitím viacvláknovosti.

Topológia: Procesy sú rozdelené do párov (0-1, 2-3 atď.). Vnútri každého procesu sa vytvorí vyhradené vlákno na pozadí (`Receiver Thread`), ktoré nepretržite číta prichádzajúce dáta. Hlavné vlákno (`Handler Thread`) nepretržite generuje a odosiela dáta partnerovi.

Cieľ: Maximálny stresový test pamäťového podsystému. Overenie duplexnej komunikácie a bezpečnosti knižnice pre vlákna.



Na grafe priepustnosti (Bandwidth) je jasne viditeľný lineárny rast rýchlosti na logaritmickej mierke. Pri objemoch správ nad niekoľko megabajtov dosahuje priepustnosť teoretické maximá (terabajty za sekundu virtuálnej priepustnosti), čo empiricky potvrdzuje úspešnú implementáciu mechanizmu Zero-Copy. Červená čiara (asynchrónny duplex) vykazuje vyššie hodnoty vďaka efektívnemu využitiu viacjadrových procesorov a zreťazeniu (pipelining) správ.