1. semestrální úloha KIV/DS

Michal Bryxí - A08N0060P - picca@students.zcu.cz

25. října 2009

1 Zadání

Naprogramujte v jazyce C server, který bude realizovat funkci sekvenceru. Přijaté požadavky bude rozesílat skupině procesů v pořadí, ve kterém je přijal. Server bude realizován jako paralelní, bude schopen přijímat jeden nebo více požadavků najednou a vysílat více požadavků najednou.

Paralelnosti dosáhnete dynamickým spouštěním paralelních procesů (jedna varianta) nebo spouštěním více vláken (jeden požadavek, jeden proces nebo vlákno). Přenášený požadavek bude textová (binární) zpráva celkové délky 16KB slabik, kde prvních 8 slabik bude zprávu identifikovat a zbytek bude náhodná výplň. K identifikaci zprávy využijte IP adresu a port odesílatele spolu s pořadovým číslem zprávy. Formát identifikace bude tedy následující.

Každý příjemce zprávy může být i odesílatelem (skupina příjemců a odesílatelů jsou identické). Členství ve skupině je statické (vznikne na základě konfigurace ? zadání doménového jména), ne všichni členové musí být funkční. Cílem je přenést mezi členy skupiny zprávy tak, aby v každém z uzlů byly přijaty ve stejném pořadí. K přenosu použijte TCP spojení, které bude sloužit k přenosu pouze jedné zprávy (vytvoří se spojení mezi uzlem a sekvencerem na vyhrazeném portu, přenesou se data, zařadí se do fronty, zruší se spojení). Skupinový přenos od sekvenceru k uzlům bude realizován vícenásobně dvoubodovým spojem. Problém volby přijímacího portu v případě, že na jednom uzlu bude více přijímacích procesů řešte zadáním čísla naslouchacího portu do konfigurace.

Součástí řešení bude i měření časů na sekvenceru. Budete měřit dobu t1 přenosu zprávy od uzlu do sekvenceru, dobu t2 čekání zprávy (ve frontě) na odeslání, dobu t3 odeslání zprávy všem příjemcům (i odesílateli!) ? tedy od započetí vysílání prvnímu příjemci po ukončení spojení s posledním příjemcem.

K ověření správnosti algoritmu je třeba nastavit v jednotlivých přijímacích procesech (vláknech) různá zpoždění, aby byla data přenášena různými rychlostmi (sekvencer musí pro daný uzel počkat s posíláním další zprávy dokud nebude dokončen přenos předchozí ? v přijímacích uzlech není fronta! Omezení rychlosti jednoho uzlu by nemělo omezovat rychlost komunikace sekvenceru s ostatními uzly (některé přenosy mohou mít náskok před jinými)! Nevyváženost rychlostí by se měla projevit pouze narůstáním fronty se zprávami v sekvenceru. Pokud použijete paralelní procesy, je třeba si uvědomit, že fronta zpráv musí být ve sdílené paměti. V každém případě je třeba zajistit synchronizaci tak, aby doba odezvy systému byla co nejkratší při splnění podmínky sekvenčního doručování zpráv do jednotlivých uzlů.

Před odevzdáním ověřte funkčnost Vašeho produktu pro konfiguraci dvou uzlů se 3 procesy (vlákny) na každém z nich. Sekvencer bude spuštěn jako čtvrtá úloha na jednom z uzlů.

2 Řešení

2.1 Klient

Po spuštění klienta dojde k zkontrolování zadaných údajů a následnému připojení na server. Hlavní vlákno periodicky posílá zprávy na server. Druhé vlákno pak poslouchá na nastaveném portu a v případě že dostane zprávu tuto okamžitě vypíše na standardní výstup.

Formát zprávy je zvolen jako *[port klienta]#[zpráva]* . Jelikož mi přišlo rušivé posílat / vypisovat opravdu náhodné znaky rozhodl jsem se pro posílání 16k stejných znaků. Na funkci to nemá vliv a s výhodou toto lze použít ve výpisech Sequenceru, jež jsou poté přehlednější.

2.2 Server / Sequencer

Mnohem zajímavějším problémem je pak samotný sequencer. Server po spuštění také zkontroluje parametry a začne poslouchat na daném portu. Druhé, odesílající vlákno ve smyčce kontroluje obsah fronty zpráv na odeslání.

Po přijetí zprávy od klienta je rozparsována zpráva na část informační a část ve které je port. Dále se zjistí IP adresa klienta. Pokud klient ještě nekontaktoval server, bude zařazen do seznamu klientů. Zpráva je podle konfiguračního souboru a podle portu na které klient poslouchá zařazena do jedné z front.

Ve chvíli kdy se vzbudí vlákno, jež periodicky kontroluje frontu pro odeslání a zjistí že ve frontě je nová zpráva provede toto vytvoření nového vlákna, jenž tuto zprávu odešle.

Server vypisuje ladící výpisy ohledně doby odesílání zpráv a obsahu zpráv. Server nevypisuje celé zprávy, ale pouze jeden znak jímž je celá zpráva tvořena. Pro zajištění bezpečného přístupu k frontě zpráv byl použit mutex.

3 Závěr

Logika aplikace je poměrně přímočará. Princip sekvenceru není nic složitého. Jediné na co si člověk musí dát pozor je vzájemné neblokování jednotlivých částí aplikace a opatrný přístup / implementace fronty odesílaných zpráv. Odevzdaná aplikace doufám splňuje zadání. Použití jazyka *C* celou aplikaci poměrně zkomplikovalo. Použití vyššího programovacího jazyka by bez nadsázky vedlo k 10% řádků kódu. Některé konstrukce v tomto jazyce mi prostě nesedí. Například oficiální postup přiřazení *int*u do *string*u, který jsem našel na internetu je opravdu nepěkný. Nicméně je potěšující, že jsem aplikaci dokázal sestavit i v tomto jazyce.