P1- Client Server

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P\_R | P\_W | Δx (s) | Δt (ms) | Timp (ms) |
| 4 | 4 | 1 | 1 | 42001.0 |
| 2 | 41881.0 |
| 4 | 41929.0 |
| 2 | 1 | 81962.0 |
| 2 | 82165.0 |
| 4 | 83076.0 |
| 2 | 2 | 1 | 1 | 41882.0 |
| 2 | 41881.0 |
| 4 | 42743.0 |
| 2 | 1 | 83653.0 |
| 2 | 82760.0 |
| 4 | 83133.0 |
| 4 | 2 | 1 | 1 | 41607.0\* |
| 2 | 41633.0 |
| 4 | 41783.0 |
| 2 | 1 | 83095.0 |
| 2 | 81873.0 |
| 4 | 82706.0 |
| 4 | 8 | 1 | 1 | 41779.0 |
| 2 | 41723.0 |
| 4 | 41639.0 |
| 2 | 1 | 81812.0 |
| 2 | 81719.0\* |
| 4 | 81774.0 |

Cazurile de testare:

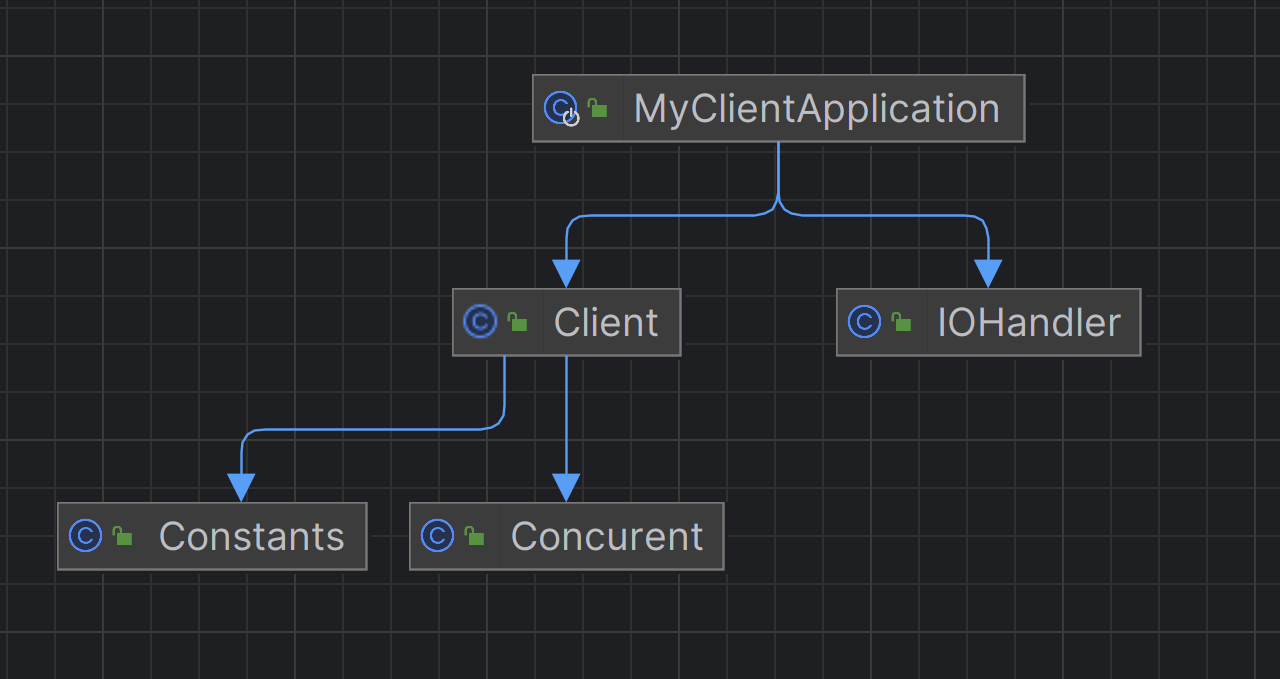
Pentru Δx = 1, cazul cu p\_r = 4, p\_w=2 si Δt=1 se dovedeste a fi cel mai eficient (41607.0ms). Acest lucru poate fi atribuit eficienței echilibrate între viteza de procesare a cererilor de la clienți (cititorii) și capacitatea de a actualiza informațiile (scriitorii). Cu 4 cititori, serverul poate prelua date rapid de la mai mulți clienți simultan, în timp ce 2 scriitori sunt suficienți pentru a gestiona coada de actualizări fără a crea blocaje, mai ales când intervalele de timp între trimiterea loturilor de date (Δx) și timpul de răspuns (Δt) sunt mici. În comparație cu alte configurații cu Δx = 1, acest scenariu echilibrează eficient resursele fără a suprasolicita sistemul.

Pentru Δx = 2, cazul cu p\_r = 4, p\_w=8 si Δt=2 se dovedeste a fi cel mai eficient (81719.0ms). Având mai mulți scriitori decât în celelalte cazuri cu Δx = 2, permite serverului să gestioneze și să actualizeze coada globală de concurenți mai eficient. În plus, intervalul mai mare de Δt permite un timp de răspuns suficient pentru ca serverul să proceseze cererile fără a fi nevoit să recalculeze clasamentele prea des, reducând astfel timpul total de execuție. Această configurație oferă un echilibru între numărul mare de scriitori care procesează rapid actualizările și intervalul de timp care minimizează recalculările, ducând la o eficiență sporită.

Concluzia analizei este că eficiența aplicației client-server în procesarea și răspunsul la cereri este afectată semnificativ de configurarea numărului de thread-uri pentru cititori și scriitori, precum și de intervalul de timp între trimiterea datelor și intervalul de timp de răspuns.

Rezolvarea problemei se bazează pe implementarea unei aplicații client-server care gestionează un concurs internațional. Fiecare client reprezintă o țară și trimite rezultatele concurenților săi către server în loturi, procesate apoi într-un sistem cu thread-uri pentru citire și scriere. Logica din spatele implementarii operatiilor pe coada si lista este aceeasi ca si in labul anterior. De asemenea si strusturile de date folosite persista si in acest caz. Serverul organizează datele primite într-o coadă și calculează clasamentul țărilor, răspunzând la cererile clienților. Performanța este testată prin ajustarea numărului de thread-uri și a intervalului de timp pentru cereri și răspunsuri. Implementarea folosește concepte de programare concurentă, inclusiv future/promises și thread pools, pentru a îmbunătăți eficiența.

Client diagram



Server diagram

