|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **작성자** | **2014132002 구태균** | **팀명** | - |
| **주차** | **17 주차** | **기간** | **2020.05.04 ~ 2020.05.10** | **지도교수** | **정 내 훈** (서명) |
| **이번주 한일** | * 16주차 Runtime Error 해결 * 벤치마크 프로그램 구현 | | | | |

1. **16주차 Runtime error 해결**

**[원인] enable\_shard\_from\_this에서 control\_block release()와 weak\_release()에서 중복해서 delete this 명령어를 수행**

**[해결] Release()에서 enable\_shrared\_from\_this의 경우와 아닌경우를 나누어 delete this를 수행.**

1. **벤치마크 프로그램 구현**

**<게으른 동기화>**

**lfsp::shared\_ptr error**

**std::shared\_ptr / atomic\_load() , atomic\_store()**

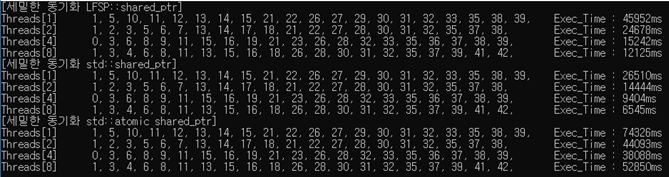
* **비교 불가**

**<세밀한 동기화>**

**lfsp::shared\_ptr**

**std::shared\_ptr**

**std::shared\_ptr / atomic\_load() , atomic\_store()**

****

1. **벤치마크 프로그램 Error**

**3.1 std::shared\_ptr / lfsp::shared\_ptr 비교 결과, 함수 인자로 const의 사용이 불가능.**

* **[게으른 동기화] validate()의 인자로 (const lfsp::shared\_ptr& s) 불가능 확인.**

**3.2 [게으른 동기화] list와 shared\_ptr의 lock으로 정상적으로 구현되지 않음.**

**container에서의 lock은 Node를 추가할 때 관련되지만, share\_ptr의 lock은 ctr,ptr을 관리.**

* **[게으른 동기화] Debug & Release mode 싱글코어 정삭작동, 멀티코어 비정상 작동 (빈번히 정상)**

**Thread A, B가 동작 중, A에서 추가된 Node가 B에 atomic하게 동작하지 않음.**

**curr = curr->next(shared\_ptr& operator=(shared\_ptr&))에서 shared\_ptr의 lock이 이용되는데, list에서 Node가 추가되면 (shared\_ptr& operator=(shared\_ptr&))중간에 데이터 변경**

**(데드락x, error : unlock of unowned mutex)**

**[원인 1] list와 shared\_ptr의 두종류의 lock으로 atomic하지 않은 구현.**

**[해결방안 1] Node를 추가할 때, shared\_ptr의 lock()을 시도.**

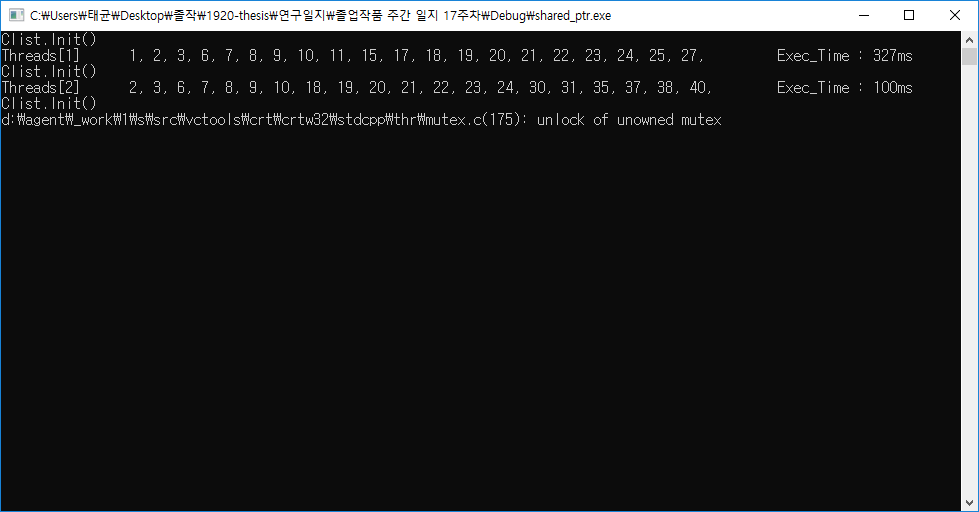
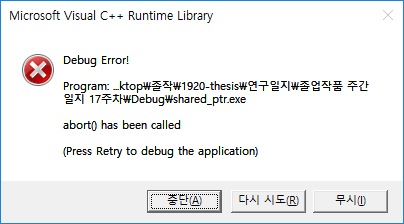
* **2종류의 lock을 쓰면 성능은??**

**[원인 2] 메모리 일관성**

**[해결방안 2] atomic\_thread\_fence와 volatile 사용.**

* **정확한 위치 찾기가 어려움.**

**\*\* Clist.init()을 해도 Clist의 모든 Node가 해제되는건 아님. (std::shared\_ptr / lfsp::동일)**



|  |  |  |  |
| --- | --- | --- | --- |
| **문제점 정리** |  | **해결 방안** |  |
| **다음 주차** | **18 주차** | **다음 기간** | **2020.05.11 ~ 2020.05.17** |
| **다음주 할 일** |  | | |
| **지도교수**  **Comment** |  | | |