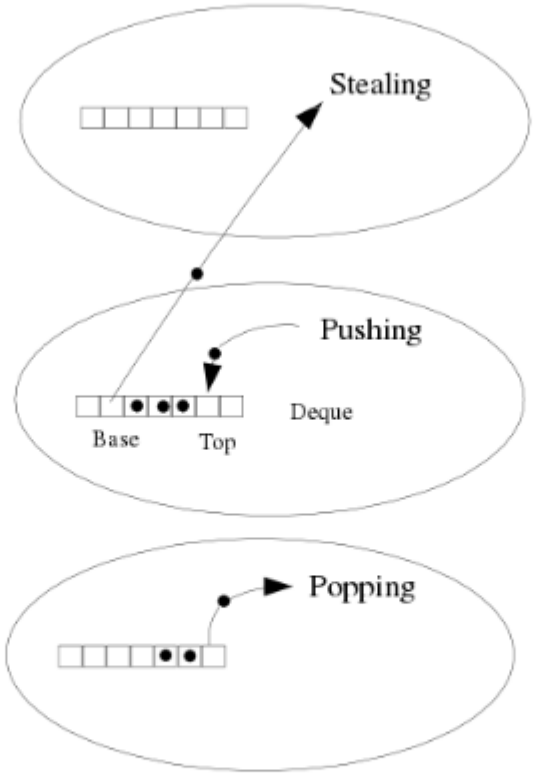
Fork/Join

Schemat problemu

```
Result solve(Problem problem) {
  if (problem is small)
    directly solve problem
  else {
    split problem into independent parts
    fork new subtasks to solve each part
    join all subtasks
    compose result from subresults
  }
}
```

ThreadPool

- zarządzanie zadaniami pomiędzy wątkami
- każdy wątek ma swoją listę zadań (worker queue)
- zadanie należy do wątku, w którym zostało utworzone
- work-stealing



źródło: Doug Lea, A Java Fork/Join Framework, State University of New York at Oswego

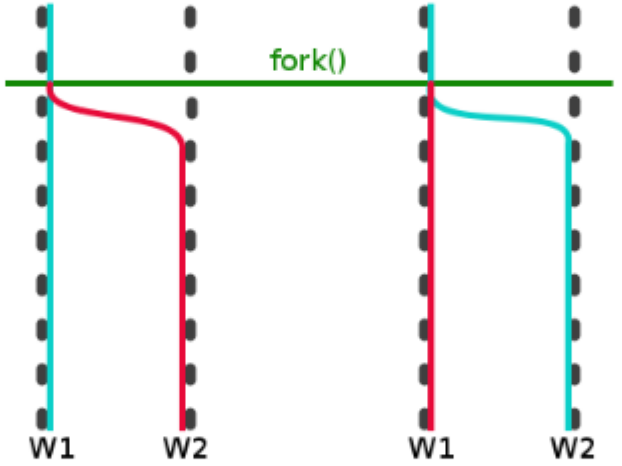
work-stealing

Child stealing - wątek, który wykonywał zadanie wywołujące fork() kontynuuje je oddając nowo utworzone zadanie do listy zadań

Continuation stealing - wątek, który wykonywał zadanie wywołujące fork() oddaje je do listy zadań, a sam wykonuje nowo utworzone zadanie

Child stealing - wątek, który wykonywał zadanie wywołujące fork() kontynuuje je oddając nowo utworzone zadanie do listy zadań

Continuation stealing - wątek, który wykonywał zadanie wywołujące fork() oddaje je do listy zadań, a sam wykonuje nowo utworzone zadanie



Co po join()?

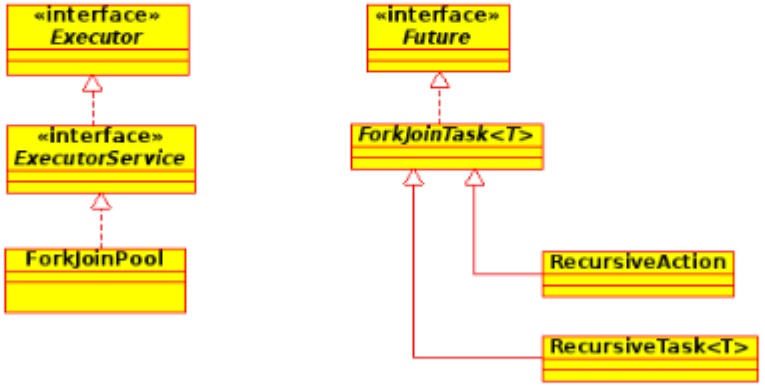
Stalling - wątek, który wywołał fork(), będzie kontynuował pracę po wywołaniu join()

Greedly - ostatni wątek, który wywoła join() będzie kontynuował pracę

Zastosowanie w aplikacjach

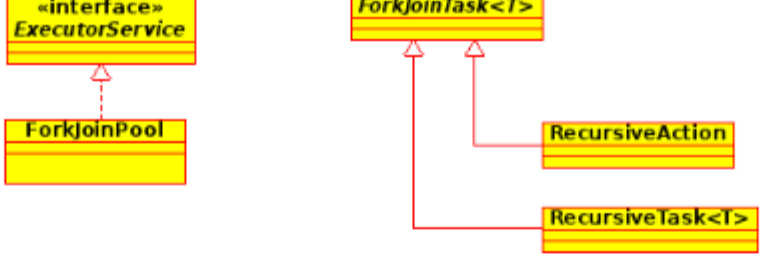
- Połączenia z zewnętrznymi usługami
- Blokady i semafory
- Zużycie pamięci

Fork/Join Framework



Główne metody ForkJoinPool:

- execute() – wykonuje zadanie asynchronicznie



Główne metody ForkJoinPool:

- execute() – wykonuje zadanie asynchronicznie
- invoke() – wykonuje zadanie synchronicznie
- invokeAll() – wykonuje kilka zadań synchronicznie

Główne metody ForkJoinTask:

- fork() – wykonuje zadanie asynchronicznie

- invoke() – wykonuje zadanie synchronicznie
- invokeAll() – wykonuje kilka zadań synchronicznie

Główne metody ForkJoinTask:

- fork() – wykonuje zadanie asynchronicznie
- invoke() – wykonuje zadanie synchronicznie
- invokeAll() – wykonuje kilka zadań synchronicznie
- join() – oczekuje na zakończenie zadania i wraca jego wynik

Dziękuję za uwagę

Adam Chyła