Выбранный подход

Был выбран подход, позволяющий распараллелить выполнение задачи без использования разделяемых ресурсов и локов. Отстутствие локов и разделяемых переменных (а следовательно и отсутствие false sharing'а и data race'ов) значительно ускоряет выполнение программы. Плюс данного подхода еще в том, что он значительно упрощает написанную программу.

Идея заключается в том, чтобы распараллелить каждую волну поиска в ширину. То есть каждая волна поиска начинается с того, что мы аггрегируем результаты предыдущей, фильтруем только те, которых в $HashSet < string > visited$ не было, кладем их в $visited$ и запускаем параллельно проход по всем этим ссылкам. При этом заметим, что нам не нужно при каждом выполнении скачивания страницы смотреть в $visited$, и брать дорогой лок на эту операцию.

Также выигрыш в производительности достигается за счет того, что в веб графах обчыно очень много ссылок, поэтому на каждой волне поиска в ширину можно выиграть много времени при распараллеливании.

Для распараллеливания загрузки страниц используется метод C# $Parallel.ForEach$. В него можно передать максимальное количество потоков, которые он может задействовать или передать -1, чтобы он он сам подобрал оптимальное количество потоков. Под капотом этот метод использует встроенную библиотеку $TaskParallelLibrary(TPL)$, а конкретно механизм пула потоков. Метод $Parallel.ForEach$ также определяет оптимальное количество Task'ов (то есть он не будет ставить 100 тысяч тасок, а объединит их в батчи и уменьшит тем самым накладные расходы).

Тестирование производительности

Производительность тестировалась на следующих настройках: crawler.exe "http://yandex.ru 2 100 crawl "При тестировании алгоритм запускался 10 раз, затем считалось среднее и дисперсия. Получились следующие результаты (результат в секундах, в скобках указана дисперсия):

1. WebCrawlerSync(): Elapsed: 55.27305424 (6.22984472886608)
2. WebCrawlerAsync(-1): Elapsed: 22.89990244 (4.26477578656015)
3. WebCrawlerAsync(1): Elapsed: 55.94903001 (1.82260854321946)
4. WebCrawlerAsync(2): Elapsed: 34.44340731 (3.26883298908731)
5. WebCrawlerAsync(4): Elapsed: 24.92590704 (4.02489514486327)
6. WebCrawlerAsync(8): Elapsed: 20.08012662 (3.41057027656706)
7. WebCrawlerAsync(16): Elapsed: 19.41109792 (2.9251830158163)

Здесь WebCrawlerSync() - полностью синхронная реализация, WebCrawlerAsync(n), где n - количество потоков, - распараллеленая реализация

Также были произведены запуски на других сайтах с большим количеством страниц с такими настройками: "crawler.exe http://google.com 3 1000 crawl_google "со следующими результатами:

1. WebCrawlerSync(): Elapsed: 806,3208642 (13,0145307000025)
2. WebCrawlerAsync(-1): Elapsed: 119,30601245 (7,22637694999998)

Из этого видно, что намного больший выигрыш в производительности можно получить на большом количестве страниц.