ИССЛЕДОВАНИЕ РАБОТЫ МНОГОПОТОЧНЫХ ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ QTHREADS НА СОВРЕМЕННЫХ МНОГОЯДЕРНЫХ СИСТЕМАХ

В.В. Корнеев, П.В. Павлухин

Создание приложений с высокой производительностью для современных многоядерных процессоров требует понимания особенностей работы с общими для множества ядер ресурсами, такими как L3-кэш, ввод-вывод, сетевой доступ, поэтому важной становится задача динамического планирования разделяемых ресурсов для мультитредовых приложений. Число ядер в одном вычислительном узле на сегодняшний день достигает нескольких десятков. Такие «тяжелые» узлы требуют от программиста учета специфики не только межузлового параллелизма, но и параллелизма внутри узла, особенно для задач с нерегулярными и/или адаптивными решениями. Учитывая увеличивающуюся неоднородность в аппаратном обеспечении, становится все сложнее выполнять эффективное отображение мультитредовых программ на доступные ресурсы (вычислительные ядра); конкуренция за разделяемые аппаратные ресурсы (например, L3-кеш) между тредами приводит к появлению конфликтов и узких мест. В связи с этим растет необходимость в runtime-системе для мультитредовых приложений, с высокой производительностью работающей на многосокетных вычислительных узлах и способной наиболее эффективно задействовать общие ресурсы. При этом крайне желательно, чтобы данная runtime-система могла использоваться в одной из распространенных моделей многопоточного программирования.

В докладе будут представлены результаты исследования одной из таких runtime-систем, поставляемой вместе с библиотекой Qthreads [1, 2, 3]. Данная библиотека предназначена для написания приложений с общей памятью с использованием «легковесных» тредов – qthread-ов. В отличие от pthread-ов, qthread-ы имеют намного меньший стек, переключение между ними выполняется в пространстве пользователя, а не через системный вызов, а для синхронизации используются т.н. full-empty биты. Одно из достоинств данной библиотеки – использование не только собственного API для написания программ, но и возможность компилировать с ее помощью уже существующие OpenMP-приложения без модификации исходного кода. Runtime-система Qthreads использует несколько видов планировщиков для динамической балансировки нагрузки и эффективного использования общих процессорных L3-кэшей. В результатах исследования будут представлены измеренные характеристики runtime-системы (время переключения между qthread-ами, накладные расходы на балансировку нагрузки и др.) и проведено сравнение с различными реализациями библиотек OpenMP, поставляемых с компиляторами.

В качестве OpenMP-приложения был выбран тест FFT, входящий в пакет Barcelona Openmp benchmarks. Для начала была выполнена трансляция исходного кода теста с помощью Rose compiler. Этот source-to-source транслятор заменяет OpenMP-прагмы на соответствующие вызовы функций из библиотеки qthreads через xomp-интерфейс. Полученный в итоге модифицированный код компилировался стандартным образом с линковкой библиотеки qthreads. В качестве планировщика q-тредов использовался mts-scheduler (sherwood) из проекта MAESTRO, который создает shepherd-ы по одному в каждом numa-домене. Shepherd представляет из себя привязанные к отдельным ядрам "тяжелые" p-треды (их число равно кол-ву ядер в домене) с общей локальной очередью q-тредов,которые динамически раскидываются на закрепленные p-треды. В случае опустошения локальной очереди происходит захват части q-тредов из очереди другого shepherd-a (work-stealing). Данный планировщик показывает свою эффективность прежде всего на задачах с прагмами task, когда число openmp-нитей заранее неизвестно и/или выполняется рекурсивное порождение нитей.

Выполнялся замер времени работы 3 версий FFT с библиотечными реализациями OpenMP gcc (4.6.2), icc (12) и qthreads(gcc) (1.8.1). Тестирование проводилось на 2-сокетном узле с AMD Opteron 6274, для gcc- и icc-версии привязки тредов к ядрам задавались при запуске задачи. Как видно из Таблицы 1, реализация openmp-библиотеки в gcc не дает масштабирования счета после 8 ядер, icc- и qthreads-версии дают близкие результаты. Дополнительно были проведены тесты на 8-сокетном узле с 8 10-ядерными Intel Xeon E7-8870. Сранивались версии с библиотеками OpenMP и qthreads на компиляторе Intel. Версия с gcc была исключена из-за проблем с масштабированием. В диапазоне 10 — 50 ядер (Таблица 2) OpenMP-версия оказывается производительнее, однако при более полном задействовании доступных ресурсов qthreads-версия приближается к первой, даже опережая ее на 60 и 80 ядрах.

Таблица 1. Время выполнения теста FFT на узле с 2 x AMD Opteron 6274

Число ядер	1	4	8	16	24	32
gcc,c	18,9	8,4	5,4	10,9	16,9	22,6
qthreads, c	18,9	7,9	4,8	2,9	1,92	1,53
icc, c	18	7,4	4,3	2,9	1,96	1,52

Таблица 2. Время выполнения теста FFT на узле с 8 x Intel Xeon E7-8870

Число ядер	5	10	20	30	40	50	60	70	80
qthreads, c	10,65	7,64	5,36	4,6	3,46	3,31	2,76	2,77	2,4
icc, c	11,33	6,1	4,03	3,58	3,15	3,05	3,06	2,73	2,8

ЛИТЕРАТУРА:

1. K. Wheeler, R. Murphy, D. Thain "Qthreads: An API for Programming with Millions of Lightweight Threads" // In the Proceedings of the 22nd IEEE International Parallel & Distributed Processing Symposium (IPDPS '08, in the MTAAP '08 workshop), IEEE Press, 2008
2. S. Olivier, A. Porterfield, K. Wheeler, J. Prins "Scheduling Task Parallelism on Multi-Socket Multicore Systems" // In the Proceedings of the 25th International Conference on Supercomputing (ICS'11, in the ROSS'11 workshop), ACM Press, 2011
3. A. Porterfield, R. Fowler, P. Horst, D. O'Brien, S. Olivier, K. Wheeler, B. Viviano "Scheduling OpenMP for Qthreads with MAESTRO" // Technical Report TR-11-02, Renaissance Computing Institute, September, 2011