УДК 004.4'422

Поддержка гетерогенных вычислений в современных языках программирования

К.В. Юхин[1,2]

[1]Компания Интел

[2]Московский физико-технический институт (государственный университет)

Большинство современных языков программирования поддерживают концепции параллельных вычислений в весьма ограниченном виде. Языки системного программирования, такие как C и C++ не поддерживают параллельных вычислений вообще. Ориентированные на применение в научной сфере языки, например Fortran, содержат средства, позволяющие выразить параллельность в явном виде, хотя и в достаточно узком смысле. Речь здесь идет в основном об обработке массивов данных.

Приложения можно классифицировать по уровню связности подзадач. Приложения, в которых подзадачи не требуют слишком частых синхронизаций (менее нескольких раз в секунду) относятся к модели «coarse-grain parallelism», приложения с частыми синхронизациями подзадач относят к модели «fine-grain parallelism» [3].

Для того чтобы обеспечить параллельное исполнение задач, относящихся к первому классу, можно применить специализированные библиотеки, которые для создания параллельности и синхронизаций пользуются средствами многозадачных ОС. Более того, подобные библиотеки позволяют исполнять подзадачи приложения в системах с несколькими машинами, объединёнными в сеть. Примером подобного интерфейса является MPI (Message Passing Interface) [2]. Однако, несмотря на свое удобство, использование специализированных библиотек позволяет эффективно распараллеливать только «coarse-grain» программы. Основной недостаток данного подхода в применении к «fine-grain» программам заключается в том, что дополнительные расходы, связанные с созданием, управлением и синхронизацией подзадач при использовании средств операционной системы велики.

Поэтому в данный момент наиболее перспективным подходом к распараллеливанию «fine-grain» программ является использование дополнительных языковых средств, позволяющих компилятору распараллеливать подзадачи в рамках одного приложения. Для того, чтобы добавить программе выразительности и не нарушить при этом соответствующие спецификации и стандарты, современные промышленные компиляторы содержат в себе расширения базовых языков программирования. Эти расширения позволяют точнее описывать зависимости внутри программного кода. Спецификации таких расширений разрабатываются обычно независимыми некоммерческими организациями. Примером такого расширения является спецификация OpenMP [4], которая позволяет предоставлять

компилятору информацию как о зависимостях по данным, так и о структурных связях между подзадачами.

Практика показывает, что наибольшей эффективности использования вычислительных ресурсов можно достичь, применяя специализированные вычислительные единицы для конкретных подзадач. Это относится к широкому спектру вычислительных задач – от высокопроизводительных вычислений до бортовых систем. В результате вычислительная система становится гетерогенной, в ней появляются узлы, которые с разной степенью эффективности выполняют разные классы подзадач. Для того чтобы использовать подобные вычислительные системы для задач класса «coarse-grain» хорошо подходит MPI. Однако следует иметь в виду, что MPI является примером решения, требующего поддержки периода исполнения. Многие специализированные вычислительные единицы работают без операционной системы, поэтому интерфейс MPI к ним неприменим [1]. Решением здесь может быть искусственное разделение программы на подзадачи и сборка этих подзадач раздельно, с использованием специфических инструментов. Однако такой подход исключает переносимость получаемых кодов.

Для того чтобы обеспечить переносимость кодов, ориентированных на использование в гетерогенных системах, спецификация OpenMP была расширена возможностью выделения участков кода, которые должны быть исполнены на другой вычислительной единице. Процесс переноса вычислений с основного узла на специализированный узел называется выгрузкой (offload).

Целью работы была реализация поддержки выгрузки вычислений в компиляторе GCC (GNU Compiler Collection). Основным требованием к реализации было соблюдение универсальности. Реализация позволяет производить выгрузку на любые архитектуры, для которых реализован генератор кода GCC. Поддержки периода исполнения со стороны узлов, на которые производится выгрузка не требуется: вся ответственность за реализацию взаимодействия по выгрузке вычислений и загрузке результатов лежит на компиляторе.

Литература

1. *Gabriele Jost and Haoqiang Jin, Dieter An Mey and Ferhat F. Hatay*. Comparing the OpenMP, MPI, and Hybrid Programming Paradigm on an SMP Cluster // Proceedings of Fifth European Workshop on OpenMP, 2003.
2. MPI: A Message-Passing Interface Standard. [Электронный документ]. (http://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf). Проверено 10.10.2015.
3. *Polychronopoulos C.D.* Parallel Programming and Compilers. // Kluwer Academic Publishers, Inc., 1988, P. 7-37.
4. OpenMP 4.0 Complete Specifications. [Электронный документ]. (http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf). Проверено 10.10.2015.