Система внешних .NET утилит для настройки и оптимизации вычислительноёмких программ

Гризан С.А., Кривов М.А.
s_grizan@ttglib.org, m_krivov@cs.msu.su

Введение

В связи с активным развитием и широким распространением параллельных архитекур было разработано и предложено множество технологий и библиотек, позволяющих воспользоваться всей вычислительной мощью современных многоядерных процессоров и программируемых графических ускорителей. В качестве примера подобных технологий можно назвать такие решения как библиотеки *Intel Threading Building Blocks*, *Microsoft Parallel Extensions*, технологии *NVidia CUDA*, *AMD Stream* и стандарт *OpenCL*.

К сожалению, использование данных подходов значительно усложняет исходную программу и требует проведения её дополнительной настройки и оптимизации. Так, например, от правильного выбора сетки (грида) для для *CUDA*-потоков производительность приложения может увеличится или уменьшиться более чем на 20%. При этом оптимальное значение данного параметра заранее неизвестно, так для его определения необходима дополнительная информация об архитектуре используемого графического процессора.

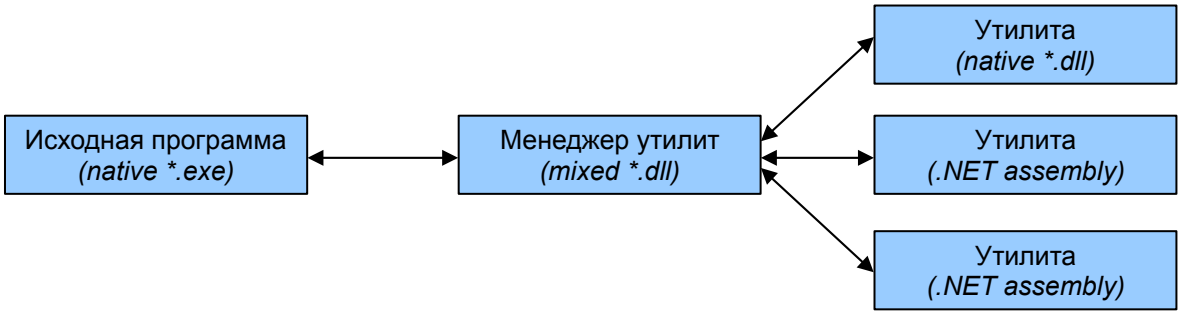
При использовании только одной из вышеперечисленных технологий подобных параметров может быть более десяти, что делает невозможным определение их оптимальных значений на этапе компиляции. В результате этого пользователи обычно задают их значения эмпирически, а недостаток производительности компенсируют либо увеличением вычислительной мощности системы, либо готовностью проведения более долгих вычислений.

Предлагаемая модель

В данной работе предлагается подход, в котором все потенциально изменяемые параметры исходной *Win32*-программы делаются «видимыми» для внешних утилит, способных их динамически изменять. Это, с одной стороны, частично решает вышеназванную проблему подбора оптимальных конфигурационных параметров, а с другой, позволяет в зависимости от промежуточных результатов менять параметры моделирования. В результате этого пользователь может сократить исходный код программы до некоторого «вычислительного ядра», вынеся управление его параметрами во внешние утилиты, и динамически подстраивать уже работающую программу под конкретную систему и решаемую задачу.

Данная идея была реализована нами в библиотеке *ttgLib* с помощью специальных шаблонных классов *Parameter<...>*. Так, к примеру, чтобы сделать произвольную целочисленную переменную «видимой» извне, её тип необходимо определить как *Parameter<int>*, после чего она может быть изменённой из внешних утилит. Стоит заметить, что реализация системы параметров в библиотеке *ttgLib* не ограничивается только рассмотренным выше механизмом — можно также создавать деревья параметров, задавать ограничения на тип (например, сделать из типа *int* интервал $[0, 42]$), обеспечивать синхронизацию и подписываться на события каждого параметра.

Взаимодействие со внешними утилитами реализовано согласно следующей схеме:



При запуске *ttgLib*-сервисов в исходной программе производится попытка найти и загрузить библиотеку *UtilManager.dll*, предоставляющую интерфейс для взаимодействия с приложением. Данная библиотека написана на языке *C++/CLI* и предоставляет как *.NET* интерфейсы, так и функции для неуправляемого кода. После этого пользователь или исходная программа может подсоединить требуемые утилиты, оформленные в виде *.NET* сборок или неуправляемых *dll*-библиотек. Если же на каком-либо этапе загрузки происходит ошибка, то утилиты будут недоступны, но исходное приложение продолжит работу.

Разработанные утилиты

На настоящий момент разрабатывается набор из следующих утилит, поставляемых вместе с библиотекой:

- *ParameterManager*. Данная утилита позволяет найти и изменить значение любого параметра, определённого в программе. Стоит отметить, что данными параметрами являются не только явно заданные пользователем переменные, но и вспомогательные внутренние параметры библиотеки *ttgLib*.
- *UsageVisualizer*. Для определения эффективности созданного параллельного приложение создана специальная утилита, проводящая визуализацию загрузки доступных вычислительных устройств. В настоящий момент поддерживается определение загрузки только центрального процессора, но планируется также добавление информации с датчиков графического ускорителя.
- *RemoteManager*. Данная утилита позволяет подключиться к работающему приложению утилитам с удалённых компьютеров, что может упростить проведение моделирования в случае, если непосредственные вычисления производится на другой машине.

Помимо этого, пользователь может разработать собственные утилиты, которые будет управлять процессом вычислений в исходной неуправляемой программе.

Дальнейшее развитие

В дальнейшем мы планируем добавление специальной утилиты, автоматизирующей процесс оптимизации приложения, использующего технологию NVidia CUDA, под конкретную вычислительную систему. Основная идея заключается в проведении серии тестовых испытаний на небольших объёмах данных, что позволит утилите попытаться подобрать набор оптимальных параметров для решения данной задачи на данной системе. После нахождения одного из локальных экстремумов функции времени выполнения, пользователь сможет проводить требуемые вычисления с помощью уже оптимизированной программы.

Другим направлением развития является реализация интерфейса взаимодействия с утилитами в виде сервиса WCF, позволяющего удаленно управлять процессом вычисления, а также отслеживать его состояние. Клиенты сервиса могут быть реализованы как на платформе .NET, так и с использованием любой другой совместимой технологии.