К анализу сложности параллельного программирования

В.Л. Авербух[1] <averbukh@imm.uran.ru>
М.О. Бахтерев[1] <mike@0xfb.imm.uran.ru>
А.Ю. Казанцев[1] <ajk.xyz@gmail.com>
В.В. Косенко[2] <vitaly.kosenko@gmail.com>

[1]ИММ УрО РАН

[2]УрГУ

V Международная научная конференция
‘Параллельные Вычислительные Технологии’
Москва, 2011

Параллельное программирование сложно!

Методы измерения качества и сложности программ.
Метрики программы:

- понятие информационного содержания программы;
- оценка необходимых интеллектуальных усилий при разработке программ;
- число требуемых элементарных решений при написании программы.

Цель:
Сравнение того насколько одна парадигма параллельного программирования сложнее или проще другой.

Параллельное программирование сложнее последовательного?

В последовательной программе существует соответствие между временем выполнения и местом выполнения. При работе с определённой строкой кода известно:

- что выполняется до неё;
- что будет выполнено после;
- к каким изменениям приведёт выполнение самой строки, и вся информация об этом содержится в ней.

Это позволяет использовать логические инварианты в рассуждениях о последовательной программе

В программе параллельной концепция места-времени разрушается.

Но в этом ли причина сложности параллельного программирования?

Аналогичные сложности, по сути, присутствуют и в последовательной программе, но, благодаря средствам программирования, ОС и среде выполнения эти сложности не видны программистам. Разработка «последовательных» программ относительно проста.

Вопросы:

- Возможна ли автоматизация в преодолении сложностей при параллельном программировании?
- Если да, то, на каких принципах ее строить?

Математический взгляд (теория следов)

Параллельность – это независимость, влекущая перестановочность: если действия a и b – независимы, то следы ($\cdot$ явно указывает конкатенацию)

$$ab \cdot \gamma \cdot aabb \cdot \delta \cdot ba \equiv ab \cdot \gamma \cdot abab \cdot \delta \cdot ab$$

эквивалентны. При создании последовательной программы мы постоянно сталкиваемся с подобными «параллельными действиями» - блоками кода, производя декомпозицию:

- программы на функции;
- памяти на занимаемые переменными и структурами данных участки;
- кода на строки, которые часто можно легко переставлять.

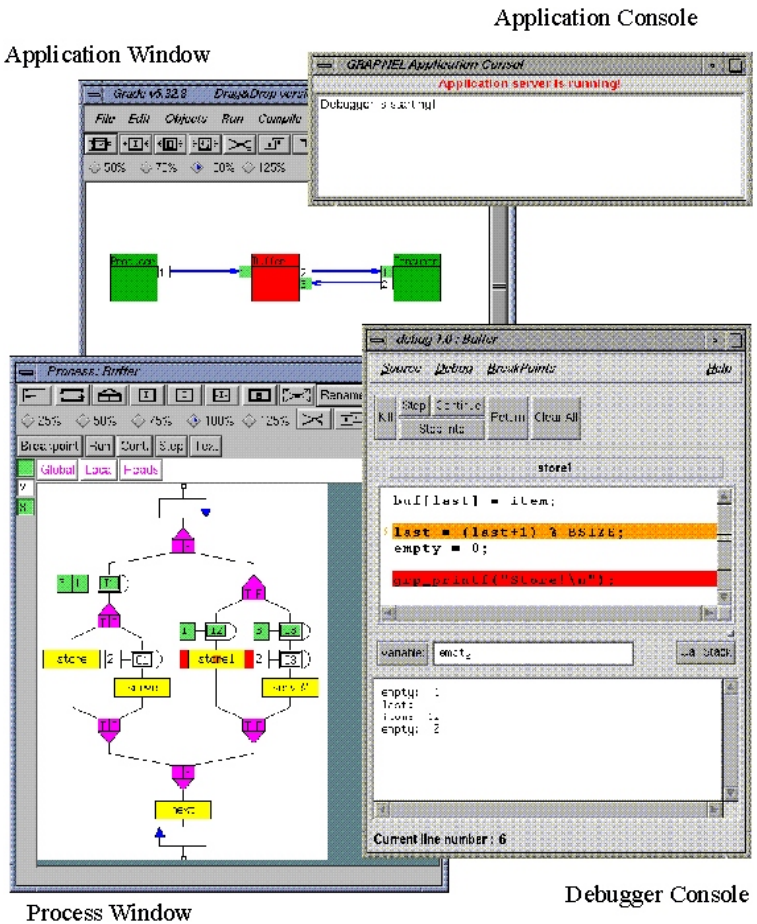
В тоже время сборка (композиция) этих блоков считается относительно простой задачей. Почему?

Секрет в данных?

Под данными здесь понимаются хранилища информации, доступ к которым в вычислительных выражениях можно задавать специальными термами. Существуют модели программирования, в которых не предполагается доступ к данным в явном виде:

- Forth
- Стековые машины
- Визуальные языки

Язык параллельного программирования Grapnel



Визуальные языки параллельного программирования

Визуальные языки, основанные на потоках данных, как правило, используют естественный способ для указания обращения к данным (стрелки).

Результат – мала выразительная сила языка – нет способа выразить сложные обращения. Сложности с визуальным редактированием. Любая перестановка блоков влечет необходимость перенаправления множества стрелок.

Анализ параллелизма

Оценка уровня связи операторной части программы с данными.
Разные инструменты могут быть предназначены для параллельного программирования машин различной архитектуры. Однако можно сравнить концепции, на базе которых построен 'интерфейс к параллельности' в совершенно различных по типу системах параллельного программирования

Анализ MPI

В MPI сообщение не является данными, в обсуждаемом (адресуемом) смысле. Поэтому программисту приходиться тратить усилия на то, чтобы данные из одного процесса превращать в данные для другого.

Анализ OpenMP

OpenMP позволяет явно использовать выражения для доступа к данным, обрабатываемым параллельно, но компилятор, генерируя параллельный (многонитевый) код, достаточно сильно меняет их семантику. Поэтому работа со сложными данными требует сложной подстройки под образ действия OpenMP-компилятора

Другие системы

Существуют системы, в которых доступы к информации, подвергаемой параллельной обработке, можно указывать в вычислительных выражениях. Термы, задающие такие доступы, имеют хорошо определённую и несложную семантику. Можно предполагать, что подобные системы существенно упрощают разработку параллельных приложений.

Примеры:

- T-Система,
- RiDE (экспериментальный результат - сокращение объёмов кода в 5 раз по сравнению с MPI).

Заключение

Рассмотрен качественный критерий оценки сложности разработки параллельных приложений в рамках различных парадигм параллельного программирования. Возможны и другие методики исследования.

Конец

Благодарим за внимание