Зимняя школа по параллельному программированию

Применение вычислительных моделей для автоматизации оптимизирующей сборки и тестирования программ на примере установки системы LuNa

Бедарев Николай, НГУ
Прокопьева Анастасия, НГУ

29.01 — 02.02 2018 г.
г. Новосибирск

Проблема сложности тестирования и запуска численных параллельных программ на вычислительных кластерах

Процесс тестирования и запуска численных параллельных программ на суперкомпьютерах с установкой необходимого окружения и последующей сборкой отчетов о результатах является сложным и трудоемким рутинным процессом:

— Приходится выполнять много рутинных, но разнообразных действий (установка необходимого окружения, контроля завершения тестов в системе очередей кластера и т.п.)

— Конкретный набор и порядок действий приходится варьировать в зависимости от обстоятельств (требуемых тестов, типов отчетов, доступности используемого оборудования и т.п.)

Данная проблема актуальна во многих сферах науки, связанных с высокопроизводительными вычислениями, и целесообразно автоматизировать этот процесс

Существующее решение

Для решения поставленной проблемы был использован математический аппарат вычислительных моделей.

Мы разработали систему, состоящую из интерпретатора вычислительных моделей и набора вычислительных моделей, по автоматическому запуску и тестированию программ.

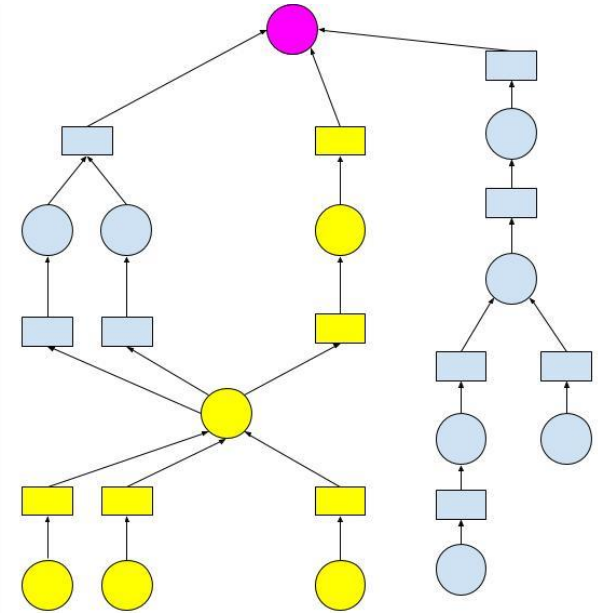
Недостатками данной системы является несовершенный синтаксис языка описания вычислительных моделей, а также малое количество вычислительных моделей

Вычислительные модели

Вычислительная модель (ВМ) — это двудольный ориентированный граф, включающий операции и переменные, связанные дугами информационных зависимостей и описывающий схему вычислений значений одних операций из других

Автоматически синтезируется план, т.е. упорядоченное множество операций, выполнение которых позволит вычислить значение заданных выходных переменных по значениям заданных входных.

Если возможны варианты, то выбирается оптимальный план по заданным критериям.



Вальковский В.А., Малышкин В.Э. ***Синтез параллельных программ и систем на вычислительных моделях****. - Издательство "Наука", Сибирское отд-е, Новосибирск, 1988г., 129 стр.*

Цель работы

Целью работу на данную школу является доработка существующей системы, путем:

- ★ усовершенствования языка описания вычислительных моделей,
- ★ создание ВМ для конкретной предметной области: добавления новых отдельных операций и описания зависимостей между этими операциями для установки и запуска системы фрагментированного программирования LuNA.

Доработка синтаксиса ВМ

ВМ поддерживаю следующий синтаксис описания операций:
<<входные_переменные>>;<<выходные_переменные>>;<операция> # <множество меток>

При этом должны выполняться следующие правила:

1) Переменные перечисляются через ','
2) Выходные переменные должны содержать как минимум одну переменную
3) Множество меток может быть пустым (в этом случае допускается отсутствие знака '#')

Доработка синтаксиса ВМ. Метки

<<входные_переменные>>;<<выходные_переменные>>;<операция> **# <<множество меток>>**

Метки имеют следующий синтаксис:

… <название_метки>=<<параметр(ы)>> …

Метки могут служит для следующих целей:

1) Указание исполнителя (например exec=local, exec=ssh, exec=mpi и тд)
2) Указание параметров исполнителя (например количество ядер для mpi)
3) Указание весов команды (например time=10, memory=20) и тд

Доработка синтаксиса ВМ. Цикл

Цикл начинается со строки:

#for <name_for> **#in** <<enumeration>>**:**

<<enumeration>> - перечисление подстановок через ','

Для подстановки элемента for'а используется синтаксис:

%{[for][<name_for>]}

Цикл завершается строкой:

#endfor

Доработка синтаксиса ВМ. Цикл

Пример:

```
#for / #in in_1,in_2,in_3:
;  %{[for][/]};  echo "%{[for][/]}" >> %{[for][/]}
#endfor
```

Данный пример эквивалентен записи:

```
; in_1; echo "in_1" >> in_1
; in_2; echo "in_2" >> in_2
; in_3; echo "in_3" >> in_3
###
```

Доработка синтаксиса ВМ. Подстановки

Подстановки объявляются следующим образом:

#define <имя_подстановки> <значение_подстановки>

Для применения подстановки необходимо указать:

%{[define][<имя_подстановки>]}

Доработка синтаксиса ВМ. Подстановки

Пример

#define *MY_NAME* nickname

; %{[define][**MY_NAME**]}.txt; echo "it is my name" >> %{[define][**MY_NAME**]}.txt

Данный пример эквивалентен записи:

; nickname.txt; echo "it is my name" >> nickname.txt

###

Доработка синтаксиса ВМ. Включения

Включения объявляются следующим образом:

#include <имя_включения>

Для применения включения необходимо указать:

%{[include][<имя_включения>][<название_искомой_переменной>][arg_1]...[arg_n]}

1) Имя включения представляем из себя название файла ВМ (возможно без расширения)

1) Аргументы из себя представляют **подстановки** которые будут выполнены при создании задачи

Доработка синтаксиса ВМ. Включения

Синтаксис аргументов определён следующим образом:

%{...[<название_подстановки>=<значение_подстановки>]...}

Пример

#include <example>

;%{[include][example][out][PATH=./example][PATH2=./example2]};res;echo 'include' >>res #

Данный пример создаст **подзадачу** со следующими характеристиками:

1) Файл ВМ == example.cm
2) Выходная переменная == out
3) В файле ВМ реализованы следующие подстановки:

- **#define** PATH ./example
- **#define** PATH2 ./example2

###

Реализация

Вычислительная модель описывается в файле (*.**cm**)

Запуск осуществляется следующим образом:

```
java -jar Parser.jar run --cm example_model.cm -o out.txt -c=memory

java -jar Parser.jar run --cm example_model.cm -o out.txt --core=time
```

Реализация

Процесс установки системы фрагментированного программирования LuNA включает в себя проверку на наличие и, при необходимости, установку следующего программного обеспечения:

- C++ compiler (g++5)
- MPI library (MPICH 3)
- Boost libraries (system, threads, date_time, program_options)
- dyncall library
- bison and flex

а также скачивание и установку самой системы.

Были созданы bash-скрипты, реализующие вышеописанный процесс, которые будут использованы в вычислительных моделях.

Всего разработано 17 bash-скриптов, что будет использоваться в 5 ВМ.

Заключение

За зимнюю школу цель была достигнута: в язык описания ВМ были добавлены такие языковые конструкции как:

- for;
- define;
- include;

были добавлены отдельные операции и разработано описание связей между ними.

В дальнейшем работу планируется продолжить путем добавления новых операций и описания связей, а также дальнейшим усовершенствованием синтаксиса и алгоритма выбора оптимального сценария.

Спасибо за внимание