Автоматизация запуска программ и их тестирования на базе вычислительных моделей

Бедарев Николай, НГУ
Прокопьева Анастасия, НГУ

09.07 — 14.07 2017 г.
г. Новосибирск

Проблема

Процесс тестирования и запуска численных параллельных программ на суперкомпьютерах с 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, используемого оборудования и т.п.)

Данная проблема актуальна во многих сферах научной и IT-деятельностях, и целесообразно автоматизировать этот процесс

Аналоги

Существующие аналоги:

- Automake/Autoconf
- Qmake/Cmake
- Scons
- GNU make
- Apache Ant(утилита)
- MSBuild

Недостаток: системами не выполняется рассмотрение альтернативных вариантов действий с выбором оптимального в текущих обстоятельствах (подобную логику приходится вручную вкладывать в схему сборки)

Цель работы

Разработать алгоритм, который бы принимал на вход:
— описание отдельных операций (“атомарных шагов”), возникающих в процессе установки и тестирования (загрузить файл на сервер, скомпилировать программу и т.п.)
— описание зависимостей между операциями,
— описание свойств операций, делающих их предпочтительными в разных ситуациях,
— спецификация результата, который требуется получить,
— критерии оптимизации синтезируемого сценария

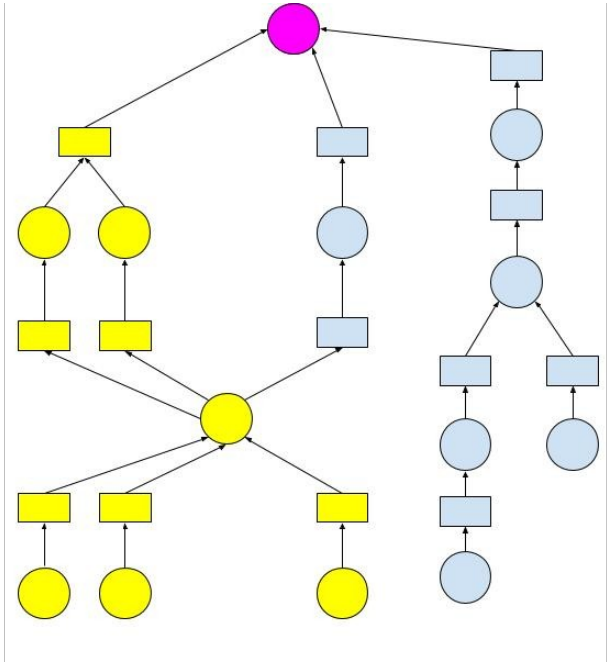
а на выход бы выдавал:
— оптимальный по заданным критериям сценарий, приводящий к заданному результату

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.



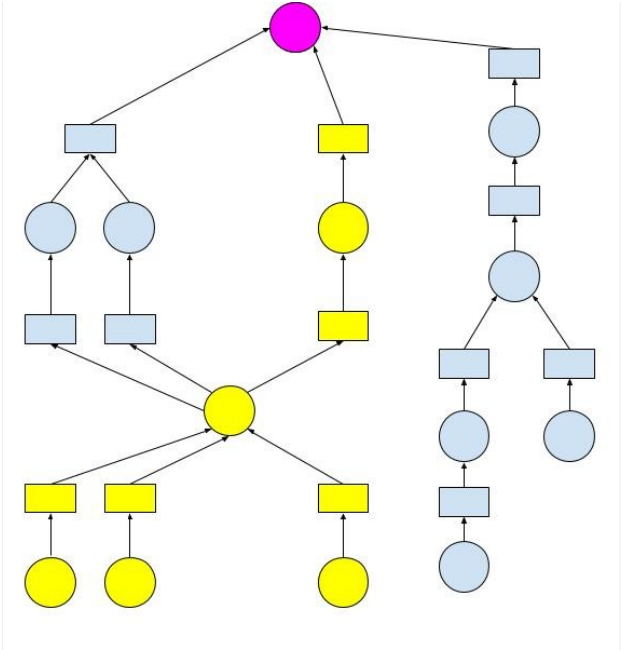
Вальковский В.А., Малышкин В.Э. ***Синтез параллельных программ и систем на вычислительных моделях****. - Издательство "Наука", Сибирское отд-е, Новосибирск, 1988г., 129 стр.*

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.



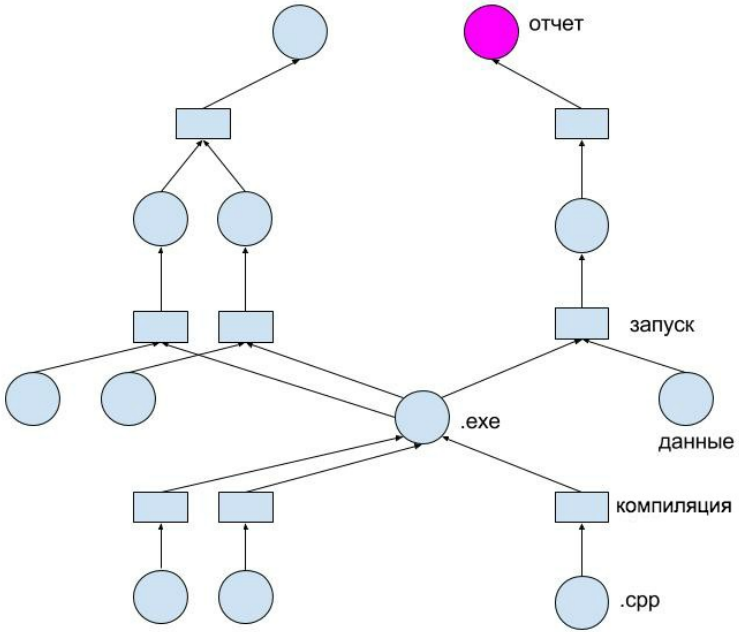
Вальковский В.А., Малышкин В.Э. ***Синтез параллельных программ и систем на вычислительных моделях****. - Издательство "Наука", Сибирское отд-е, Новосибирск, 1988г., 129 стр.*

Описание предлагаемого решения

Каждая переменная соответствует определённому состоянию процесса тестирования (как правило, соответствует получению очередной порции промежуточных файлов)

Каждая операция соответствует атомарному шагу процесса тестирования

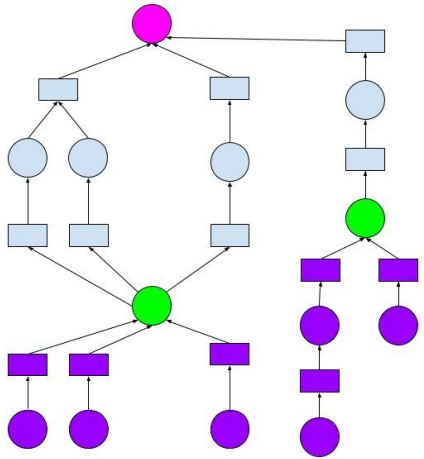
Для каждой операции задаётся набор свойств (требовательность к памяти, процессорному времени и т.п.)



Описание предлагаемого решения. Продолжение

Алгоритм (на основе алгоритма Дейкстры*):

1. Каждая операция имеет вес в соответствии с атрибутами
2. Для каждой из выходных переменных строится минимальный по весу подграф, вычисляющий её
3. Формируется общий граф операций из минимальных по весу подграфов, вычисляющих все выходные переменные



Операции итогового графа можно выполнять в порядке, не противоречащем информационным зависимостям

*) *Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн,* ***Introduction to Algorithms***

Реализация

Была реализована система, осуществляющая вывод оптимального плана и исполняющая атомарные шаги в соответствии с ним.
Язык реализации — Java.

Вычислительная модель описывается в файле (*.cm)

Запуск осуществляется следующим образом:

```
java -jar Parser.jar example_start_working.cm -o out.txt -m

java -jar Parser.jar example_start_working.cm -o out.txt -t
```

Пример вычислительной модели

```
## Пример вычислительной модели  example_start_working.cm

;file_1;touch file_1
;file_2;touch file_2

file_1 file_2;out.txt;echo "Hello world" >> out.txt
```

```
example_start_working.cm -o out.txt -t  ⇒ "Hello world"
```

Пример вычислительной модели

```
## Пример вычислительной модели  example_optimization.cm
;file_1_1;echo .>file_1_1     #time=10 memory=20
;file_1_2;echo .>file_1_2     #time=10 memory=20
file_1_1 file_1_2;out.txt;echo "example min time" >> out.txt

;file_2_1;echo .>file_2_1     #time=20 memory=10
;file_2_2;echo .>file_2_2     #time=20 memory=10
file_2_1 file_2_2;out.txt;echo "example min memory" >> out.txt
```

```
example_optimization.cm -o out.txt -t  ⇒ "example min time"
example_optimization.cm -o out.txt -m  ⇒ "example min menory"
```

Пример вычислительной модели

```
## Пример удаленного тестирования example_remotely.cm

#define USER hpcuser29@clu.nusc.ru
#define KEY /Users/Anastasia/key_nusc_school_2017.dat
#define SSH_con ssh $USER -i $KEY


;; $SSH_con gcc main.cpp -o main #mark=file_ssh
file_ssh ; result.txt ; $SSH_con ./main > result.txt
```

example_remotely.cm -o **result.txt** ⇒ "hello hpcuser29"

Пример вычислительной модели

```
## Пример удаленного тестирования с проверкой example_remotely_2.cm

#define USER hpcuser29@clu.nusc.ru
#define KEY /Users/Anastasia/key_nusc_school_2017.dat
#define SSH_con ssh $USER -i $KEY
#define SCP_con scp -i $KEY /Users/Anastasia/main.cpp $USER:~/

;;$SSH_con cat main.cpp  #result=0 mark=file_exist
;;$SSH_con cat main.cpp  #result=1 mark=file_not_exist
file_not_exist ;; $SCP_con   #mark=file_exist
file_exist;; $SSH_con gcc main.cpp -o main #mark=file_ssh
file_ssh ; result.txt ; $SSH_con ./main2 > result.txt && \
                         $SSH_con rm main.cpp && \
                         $SSH_con rm main
```

```
cm.cm                [----]  0 L:[  1+ 0   1/ 47] *(0    /2383b) 0035 0x023
# Пример вычислительной модели
# Коментарий createMark разрешены в 1 экземпляре, на строку(при корректном завершении строки)

# Получение всего необходимого для сборки луны

cm/is_gcc.sh;yesgcc;sh cm/is_gcc.sh#result=0 createMark=0=yesgcc
cm/is_gcc.sh;nogcc;sh cm/is_gcc.sh#result=1 createMark=1=nogcc
cm/is_bison.sh;yesbison;sh cm/is_bison.sh#result=0 createMark=0=yesbison
cm/is_bison.sh;nobison;sh cm/is_bison.sh#result=1 createMark=1=nobison
cm/is_boost_new.sh;yesboost;sh cm/is_boost_new.sh#result=0 createMark=0=yesboost
cm/is_boost_new.sh;noboost;sh cm/is_boost_new.sh#result=1 createMark=1=noboost
cm/is_dyncall_new.sh;yesdyncall;sh cm/is_dyncall_new.sh#result=0 createMark=0=yesdyncall
cm/is_dyncall_new.sh;nodyncall;sh cm/is_dyncall_new.sh#result=1 createMark=1=nodyncall
cm/is_flex.sh;yesflex;sh cm/is_flex.sh#result=0 createMark=0=yesflex
cm/is_flex.sh;noflex;sh cm/is_flex.sh#result=1 createMark=1=noflex

# Установка библиотеки dyncall

nodyncall cm/downloaddyncall.sh;yesdyncall;sh cm/downloaddyncall.sh#result=0 createMark=0=yesdyncall

# Сборка Луны командой LUNA

yesgcc yesbison yesboost yesdyncall yesflex;lunayes;touch lunayes
luna/Makefile lunayes;LUNA;make luna/Makefile#result=0 createMark=0=LUNA rubbish=LUNA=false

#####################################################################
# Получение test.time в зависимости от ключей оптимизации

LUNA cm/test_time.sh;test.time;time sh cm/test_time.sh > test.time#time=0 memory=100
LUNA cm/test_memory.sh;test.time;time sh cm/test_memory.sh > test.time#time=100 memory=0

#####################################################################
```

Заключение

Рассмотрена проблема автоматизации выполнения рутинных действий, возникающих в процессе тестирования параллельных программ на мультикомпьютерах

Предложен подход к автоматизации на базе вычислительных моделей

Реализован прототип системы, реализующий предложенный подход

В дальнейшем планируется переход к вычислительным моделям с массивами для возможности описывать процессы тестирования с циклически повторяющимися действиями

Спасибо за внимание

Прочие ресурсы

Адрес проекта: https://github.com/nikkollaii/parser