Новосибирский государственный университет

09.03.01 Информатика и вычислительная техника. Программная инженерия и компьютерные науки

Обеспечение специализированной поддержки конструирования фрагментированных программ для класса численных алгоритмов в системе LuNA

Выполнил студент гр. 18209 Левченко Кирилл Константинович

Руководитель ВКР: Малышкин Виктор Эммануилович, зав. каф ПВ ФИТ, д.т.н профессор

Соруководитель : Перепёлкин Владислав Александрович, н. с. ИВМиМГ СО РАН

Рецензент: Горкунов Евгений Владимирович, к.ф.-м.н., с.н.с. ИМ СО РАН

Введение в проблематику

- Разработка больших численных моделей для суперкомпьютеров требует определённых компетенций, прибегают к автоматическим системам конструирования параллельных программ.
- Система автоматического конструирования не могут построить оптимальную программу с помощью общих алгоритмов для всех задач численного моделирования.
- Целесообразно использовать частные решения для конкретного класса задач.

Актуальность

В перспективе системы автоматического конструирования будут дополняться новыми частными решениями, а также помимо интеграции частных решений, системы в перспективе будут нуждаться в реализации решений для более широкого класса задач по эксплуатации сконструированных программ, следует интегрировать множество таких решений в общую систему. Для этого система должна предоставлять удобный программный интерфейс инъекции программных модулей специализированной системы, с помощью специального модуля сборки.

Система LuNA

LuNA (Language for Numerical Algorithms) – система автоматического конструирования параллельных программ, использующая технологию фрагментированного программирования. Система LuNA является подходящей для исследования проблемы обеспечения специализированной поддержки систем конструирования параллельных программ для разных классов численных алгоритмов.

Цель работы

Разработать модуль, обеспечивающий сборку LuNA - программы из специализированных решений/системных алгоритмов.

Достижение поставленной цели требует решения следующих задач:

1. Выполнить комплексный анализ существующих частных решений, выявить их потребности для встраивания в общую систему.
2. Спроектировать программный интерфейс модуля сборки LuNA-программы.
3. Реализовать модуль сборки LuNA-программы.
4. Произвести тестирование модуля сборки LuNA-программ.

Обзор средств сборки

Решение	Преимущества	Недостатки
Compilation Pipeline	❏ разбиение на этапы ❏ разделение ответственности	❏ отсутствие поддержки вариативности сборки
Конвейер OCaml	❏ разбиение на этапы ❏ вариативность сборки	❏ не гибок для интеграции новых этапов
CI/CD	❏ автоматизация сборки, тестирование и развёртывания	❏ отсутствие поддержки вариативности сборки
Makefile	❏ вариативность сборки с помощью целей	❏ не поддерживает более сложную логику выбора этапов

Анализ требований к предлагаемому решению

Предлагаемое решение должно удовлетворять следующим требованиям:

1. Описываемые шаги сборки должны быть слабо связанным между собой.
2. Предлагаемое решение должно удовлетворять требованию вариативности.
3. Решение должно агрегировать все шаги сборки конкретного частного решения.
4. Возможность манипулировать принятием решения о выборе исполняемых шагов конкретной частной системы.
5. Модуль должен закладывать возможность исполнения одних и тех же шагов сборки на разных окружениях.

Требования к интерфейсу и реализации

Программный интерфейс модуля сборки должен соответствовать следующим требованиям:

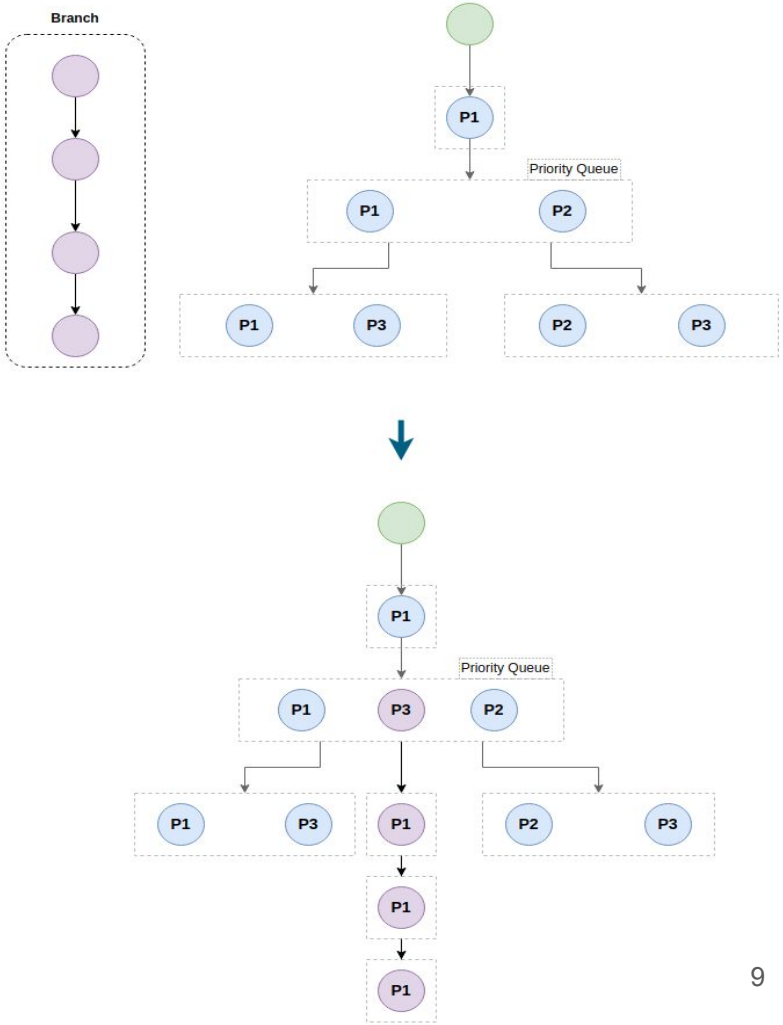
- *Инкапсуляция*
- *Изолированность*
- *Определенность*
- *Полнота*

Реализация модуля сборки должна соответствовать следующим требованиям:

- *Соответствие с парадигмой ООП*
- *Возможность конфигурировать модуль*
- *Модуль должен логировать процесс сборки*

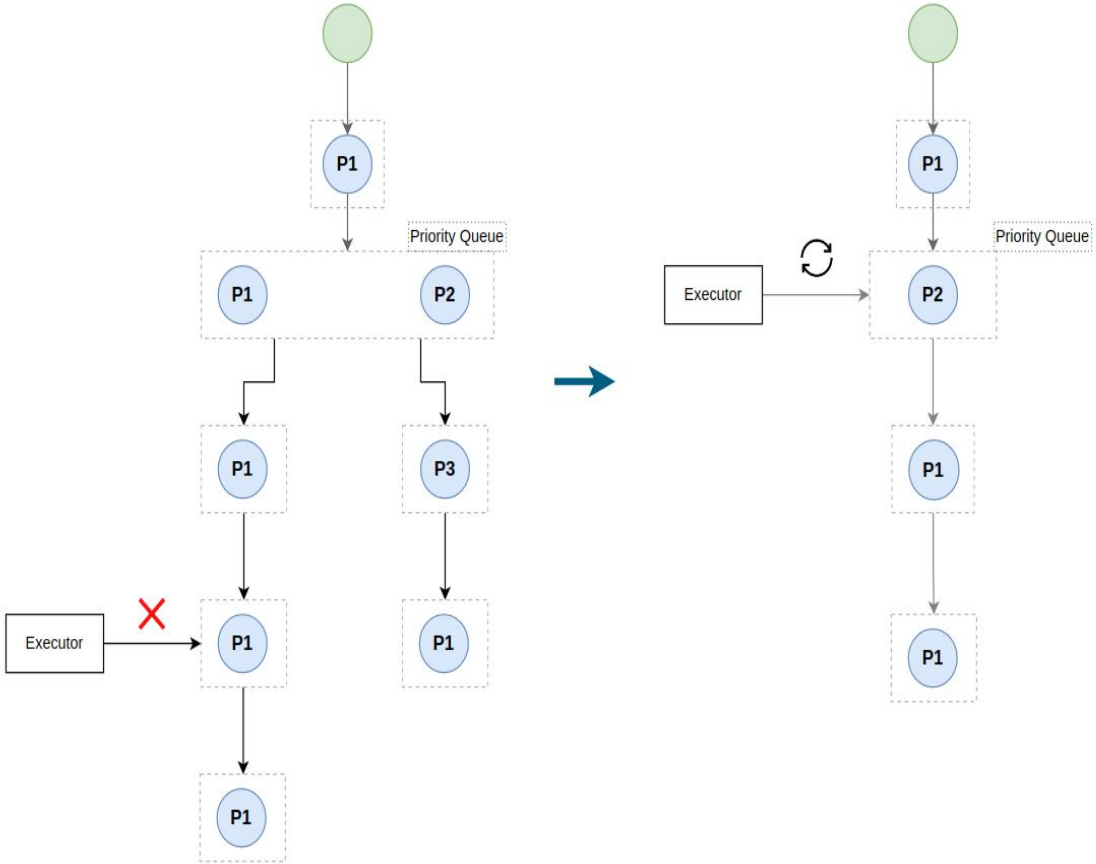
Модель процесса сборки

- ***Этап*** - отдельный шаг сборки LuNA-программы.
  - Команда для исполнения.
  - Входные зависимости.
  - Результат работы.
- ***Ветка*** - последовательность этапов, объединённая мета информацией о реализуемом частном решении.
  - Тэг - уникальный идентификатор ветки.
  - Родитель ветки.
- ***Дерево сборки*** - совокупность всех этапов, где каждый этап содержит очередь с приоритетом из дочерних этапов.



Метод обхода дерева сборки

- **Исполнитель** - объект совершающий обход дерева сборки LuNA-программы.
  - Текущий этап.
  - Журнал исполнения.
- **Окружение** - объект, ответственность которого состоит:
  - Исполнении этапа.
  - Проверки зависимостей и результатов работы этапа.
  - Удаление результатов работы этапа.
- **Операции** - у исполнителя определен набор операций над этапами и ветками.
  - Откат ветки.
  - Слияние веток.



Реализация модуля сборки

- Был реализован модуль сборки LuNA-программ
  с поддержкой специализированных решений.
- Модуль полностью реализован на языке Python с использованием следующих библиотек:
  - subprocess - исполнение этапа.
  - configargparse - конфигурация.
  - logging - журналирование.
  - PyTest - тестирование.
- Описанные ранее абстракции реализованы с помощью следующих классов:
  - Stage.
  - Requirement, Result
  - Executor.
  - Environment.

Реализация этапа

Каждый пользовательский этап переопределяет следующие методы базового класса Stage:

- *get_command()*
- *get_requirements()*
- *get_results()*

```
class GenerateCppBlocks(stage.Stage):
    def __init__(self):
        super().__init__()
        self.fcmp2 = Requirement(
            requirement=f'{cfg.luna_home}/scripts/fcmp2'
        )
            ...
        self.cpp_block_info = Requirement(
        requirement=f'{cfg.build_dir}/cpp_blocks_info.json'
        )
        self.cpp = Requirement(
            requirement=f'{cfg.build_dir}/test.cpp'
        )
        self.recommendations = Requirement(
            requirement=f'{cfg.build_dir}/program_recom.ja'
        )

    def get_requirements(self) -> [Requirement]:
        return [self.fcmp2, self.recommendations]

    def get_command(self) -> Command:
        cmd = f'{self.interpreter} {self.fcmp2} {self.recommendations}
{self.cpp} {self.cpp_block_info}'
        return Command(
            bash_command=cmd
        )
```

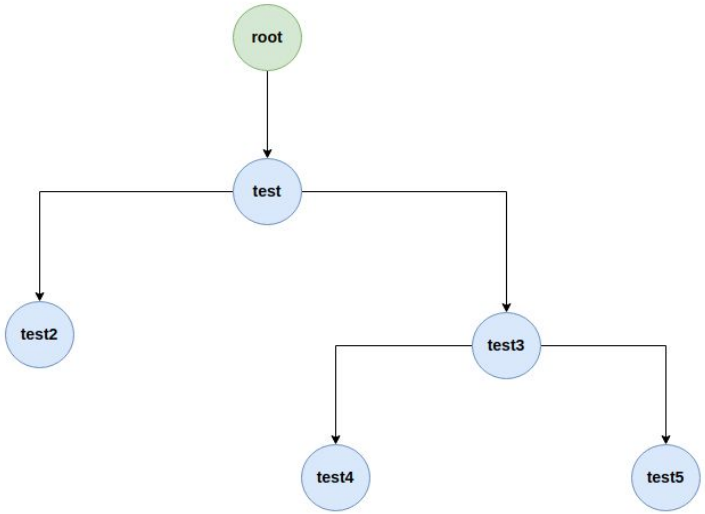
Тестирование

- **Функциональные тесты** - синтетические тесты, призванные проверить работоспособность всех функций модуля.
- Тестирование на простой программе:
  - hello world
- Тестирование на **реальных программах**:
  - Уравнение Пуассона в одномерной декомпозиции(для ветки ICLU).

```
def get_root():
    main = Branch(
        tag='main',
        stages=[
            main_branch.Preprocessor(),
            main_branch.Parser(),
            main_branch.Substitution(),
            main_branch.GenerateBlocks(),
            main_branch.GenerateRecoms(),
            main_branch.GenerateCppBlocks(),
            main_branch.GenerateMakefile(),
            main_branch.BuildLibUcodes(),
            main_branch.RunRTS(),
        ],
        environment=enviroment.LocalEnvironment('local')
    )

    iclu = Branch(
        tag='luic',
        stages=[
            iclu_branch.Preprocessor(),
            iclu_branch.Parser(),
            iclu_branch.Compiler()
        ],
        environment= enviroment.LocalEnvironment('local')
    )
```

Функциональный тест




```
============================ test session starts ============================
collecting ... collected 1 item

module_test.py::test_fully_tree PASSED                    [100%][INFO - (executor.py).Run(52) - Branch Tag: test TestStage is done.
Time: 0.0
[ERROR - (executor.py).Run(55) - Branch Tag: test2 TestStage is failed
 Status command finished with code: 1
Error: None
Message: Requirement ['notcontainsobj'] is absent
Time: 0

[INFO - (executor.py).Run(52) - Branch Tag: test3 TestStage is done.
Time: 0.0
[ERROR - (executor.py).Run(55) - Branch Tag: test4 TestStage is failed
 Status command finished with code: 1
Error: None
Message: Requirement notcontainsobj is absent
Time: 0

[INFO - (executor.py).Run(52) - Branch Tag: test5 TestStage is done.
Time: 0.0


============================= 1 passed in 0.04s ==============================
```

Тест для основной ветки исполнения

- В данном случае модуль должен попытаться собрать программу по ветке **iclu**.
- Парсер **LuNA-ICLU** не поддерживает систему рекомендаций происходит ошибка.
- Модуль сборки совершает откат до корня и собирает по основной ветке.

```
[INFO - (executor.py).Run(52) - Branch Tag: luic Preprocessor is done.
Time: 0.03479297104058787
[ERROR - (executor.py).Run(55) - Branch Tag: luic Parser is failed
 Status command finished with code: 1
Error: ERROR (line 1): invalid symbol: #
<-- here

Message:
Time: 0.0034371380461379886

[INFO - (executor.py).Run(52) - Branch Tag: main Preprocessor is done.
Time: 0.039845708990469575
[INFO - (executor.py).Run(52) - Branch Tag: main Parser is done.
Time: 0.004539916000794619
[INFO - (executor.py).Run(52) - Branch Tag: main Substitution is done.
Time: 0.05223294399911538
[INFO - (executor.py).Run(52) - Branch Tag: main GenerateBlocks is done.
Time: 0.03931760002160445
[INFO - (executor.py).Run(52) - Branch Tag: main GenerateRecoms is done.
Time: 0.05737686395877972
[INFO - (executor.py).Run(52) - Branch Tag: main GenerateCppBlocks is done.
Time: 0.04692070500459522
[INFO - (executor.py).Run(52) - Branch Tag: main GenerateMakefile is done.
Time: 0.20582091802498326
[INFO - (executor.py).Run(52) - Branch Tag: main BuildLibUcodes is done.
Time: 0.8861202570260502
coord: 102
coord: 103
coord: 101
coord: 201
coord: 202
coord: 203

[INFO - (executor.py).Run(52) - Branch Tag: main RunRTS is done.
Time: 0.3601294520194642
```

Тест для ветки iclu

- В данном тестовом случае программа является совместимой со всеми шагами сборки **LuNA-ICLU**.
- Модуль начинает и успешно заканчивает сборку и исполнение программы на ветке **iclu**.

```
[INFO - (executor.py).Run(52) - Branch Tag: luic Preprocessor is done.
Time: 0.040792019999571494
[INFO - (executor.py).Run(52) - Branch Tag: luic Parser is done.
Time: 0.010054079999463283
[INFO - (executor.py).Run(52) - Branch Tag: luic Compiler is done.
Time: 0.010981594999975641
[INFO - (executor.py).Run(52) - Branch Tag: luic MPICompiler is done.
Time: 1.314431060998686
[INFO - (executor.py).Run(52) - Branch Tag: luic Runner is done.
Time: 7.6001696519997495
DF: 4999
DF: 4999
DF: 4999
DF: 4999
```

Результаты тестирования

Исходя из результатов тестирования, можно сделать вывод, что модуль сборки показал корректную работу как на функциональных тестовых случаях, так и на реальных программах.

Заключение

- В результате работы выполнены все поставленные задачи и достигнута цель по разработке модуля сборки, что вносит вклад в задачу по обеспечению специализированной поддержки систем конструирования параллельных программ.
- **Защищаемые положения**:
  - Спроектирован интерфейс для обеспечения специализированной поддержки конструирования фрагментированных программ и реализован модуль сборки LuNA-программы.
  - Реализован модуль сборки LuNA-программ, реализующий спроектированному интерфейсу.
  - Проведено экспериментальное исследование показывающее работоспособность модуля сборки LuNA-программ.

Дальнейшие планы

- Интеграция новых частных решений.
- Расширение модуля сборки для выполнения более широкого класса задач с помощью разработанного интерфейса.

Спасибо за внимание