Лаборатория синтеза параллельных программ ИВМиМГ СО РАН

Технология фрагментированного программирования

В.А. Перепёлкин

Летняя XXXV школа-конференция по параллельному программированию
Новосибирск — онлайн, 2020 г.

Часть I. Программирующие программы, или почему программировать — не картошку копать

Программирующие программы

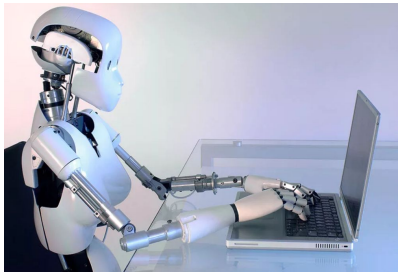
Работа программиста — это каждый раз решение новой задачи, потому что для той же самой задачи программа уже написана

А картошку приходится каждый раз копать заново

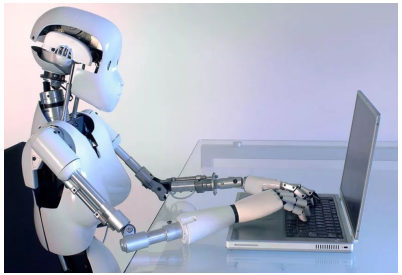
Для решения классов задач создаются инструменты

Компетенции программистов постоянно переходят в иструменты — компиляторы, утилиты, библиотеки и т.п.

Отработанные решения частных задач обобщаются, превращаясь в инструменты — программы, которые делают программистскую работу за программистов.



Для решения классов задач создаются инструменты



Решённые задачи

Нерешённые задачи

Созданием инструментов занимается системное программирование.

Самое главное — это удачное соглашение

Автоматизация программирования происходит дискретными скачками, когда создаётся новый инструмент

Инструмент — это решатель, инкапсулированный под интерфейсом (соглашением)

“Задача сводится к ...”

Пример соглашения: Java

Инкапсулированная работа: Сборка мусора, переносимость, ...

Почему стало возможно в сравнении с C++? Удачное соглашение!

- Отсутствие арифметики над указателями
- Свой байт-код и реализующая его виртуальная машина

К чему теперь сводится задача?

Часть II. Параллельно программирующие программы или технология фрагментированного программирования

ТФП: Технология Фрагментированного Программирования

Мы занимаемся системным параллельным программированием:

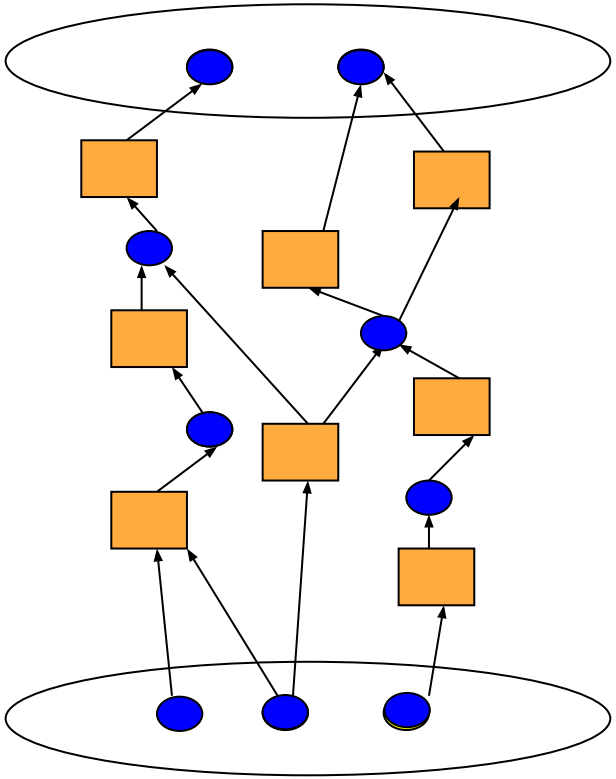
- системное программирование (создание инструментов)
- область: параллельное программирование (задач численного моделирования на суперкомпьютерах)

— Что мы хотим от ТФП?
— Не программировать параллельно, “свести задачу к...”

Соглашение: Фрагментированный Алгоритм (ФА)

Вычисления описываются как двудольный орграф фрагментов данных и вычислений (ФД и ФВ)

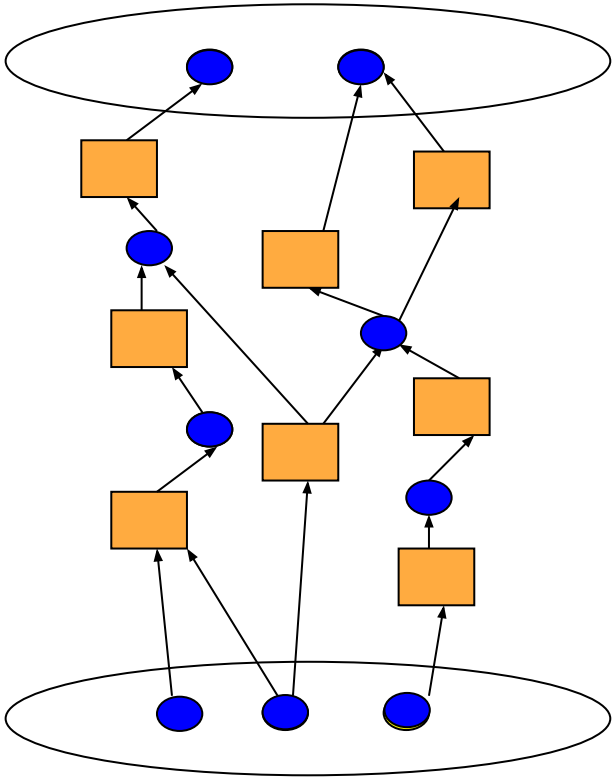
- Сериализуемость и отсутствие побочных эффектов
- Явный крупноблочный параллелизм
- Независимость от аппаратного обеспечения



Соглашение: Фрагментированный Алгоритм (ФА)

Что это даёт автоматизировать:

- Выбор управления и распределения ресурсов
- Динамические свойства: Сохранение контрольных точек, балансировка нагрузки
- ...



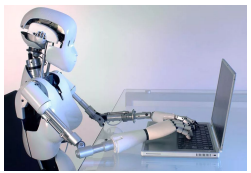
Зачем ещё рекомендации?

ФА создаёт «поле», на котором ещё нужно научиться «играть» — отображать на вычислитель

- вы не ошибетесь в управлении
- не наруите работоспособность
- «аппроксимация» параллельной программы, как конструктор

Рекомендации задают отображение на вычислитель

Сначала человек, а затем компьютер будет заниматься рекомендациями

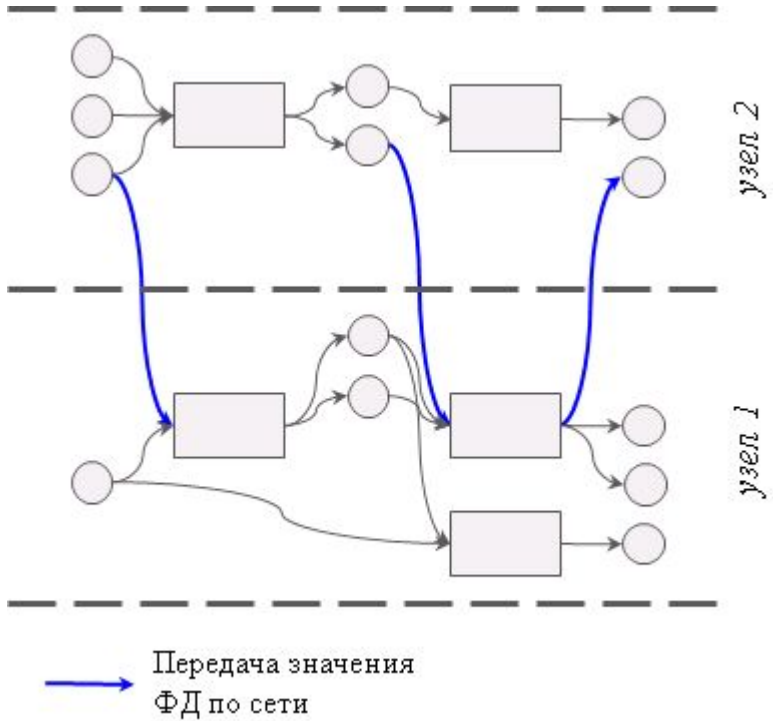


Исполнительная Система (ИС)

ИС — это распределённая виртуальная машина

Она создаёт фрагменты, передаёт по сети, запускает ФВ, хранит и удаляет ФД и т.п.

Рекомендации служат подсказками для ИС

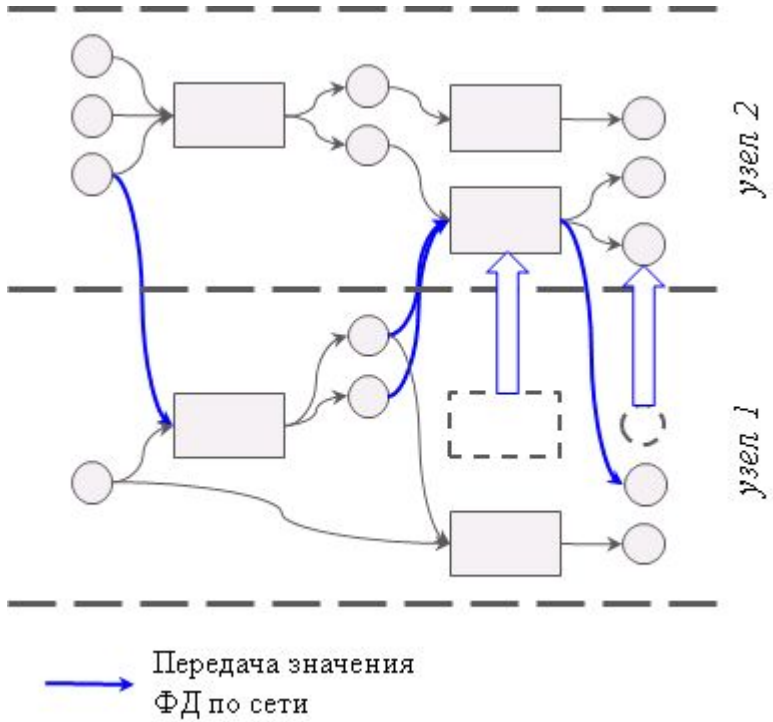


Исполнительная система

ИС — это распределённая виртуальная машина

Она создаёт фрагменты, передаёт по сети, запускает ФВ, хранит и удаляет ФД и т.п.

Рекомендации служат подсказками для ИС



А как кроме ИС?

ИС — это частный случай

Общий случай — это генерация параллельного кода с поддержкой библиотекой времени исполнения

Сейчас — это мультиагентный подход, а есть ещё генерация по шаблону

Часть III, заключительная. Технология активных знаний

Активные знания

ТФП — это как превратить ФА в программу

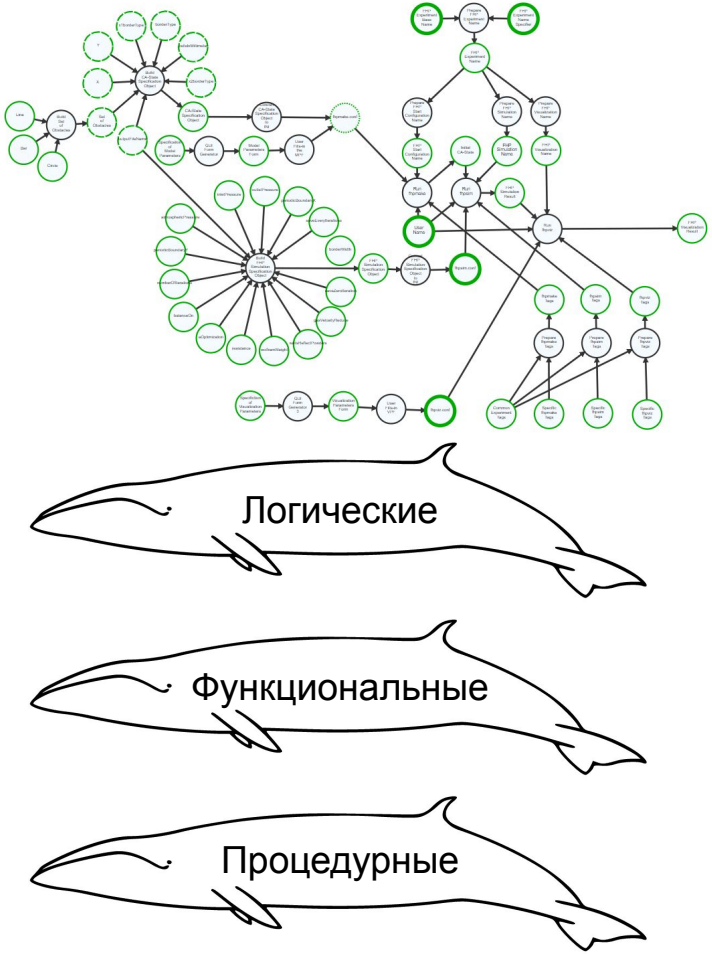
ФА — это набор *нужных* операций

Т.е. собираем вычислительное решение из операций

Задача шире — это сначала вывод ФА на основе вычислительной модели

Так мы переходим к верхнему киту — логическому программированию

ТФП — подготовка к активным знаниям.



Вопросы?