«RunTime Systems & Dataflow»

Студента 4 курса кафедры СКИ
факультета ВМК МГУ
Криволуцкого Андрея Александровича

Научный руководитель:
к.ф.-м.н. **Сальников Алексей Николаевич**

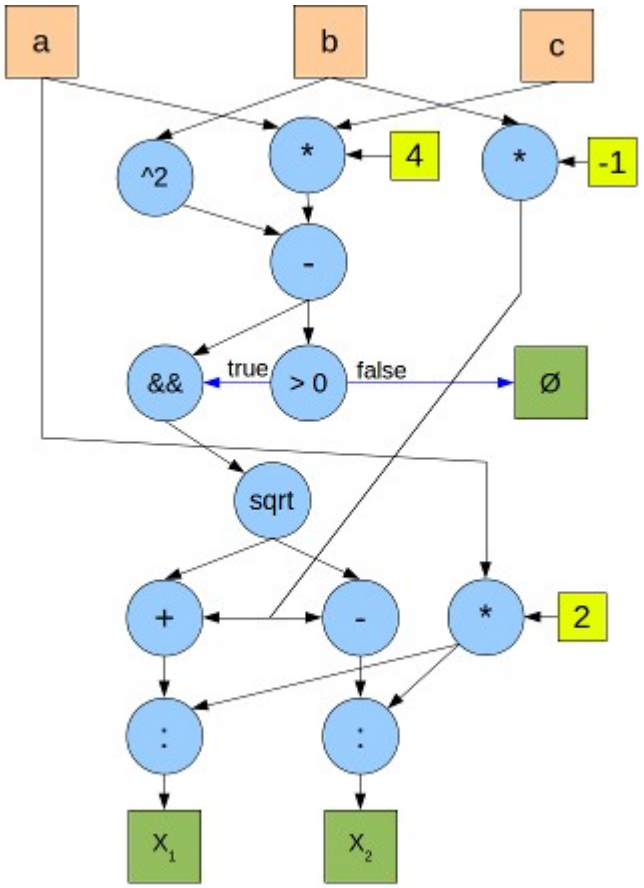
DataFlow -

Модель вычисления, управляемая потоком данных, а не потоком инструкций.

Главное преимущество — маштабируемость.

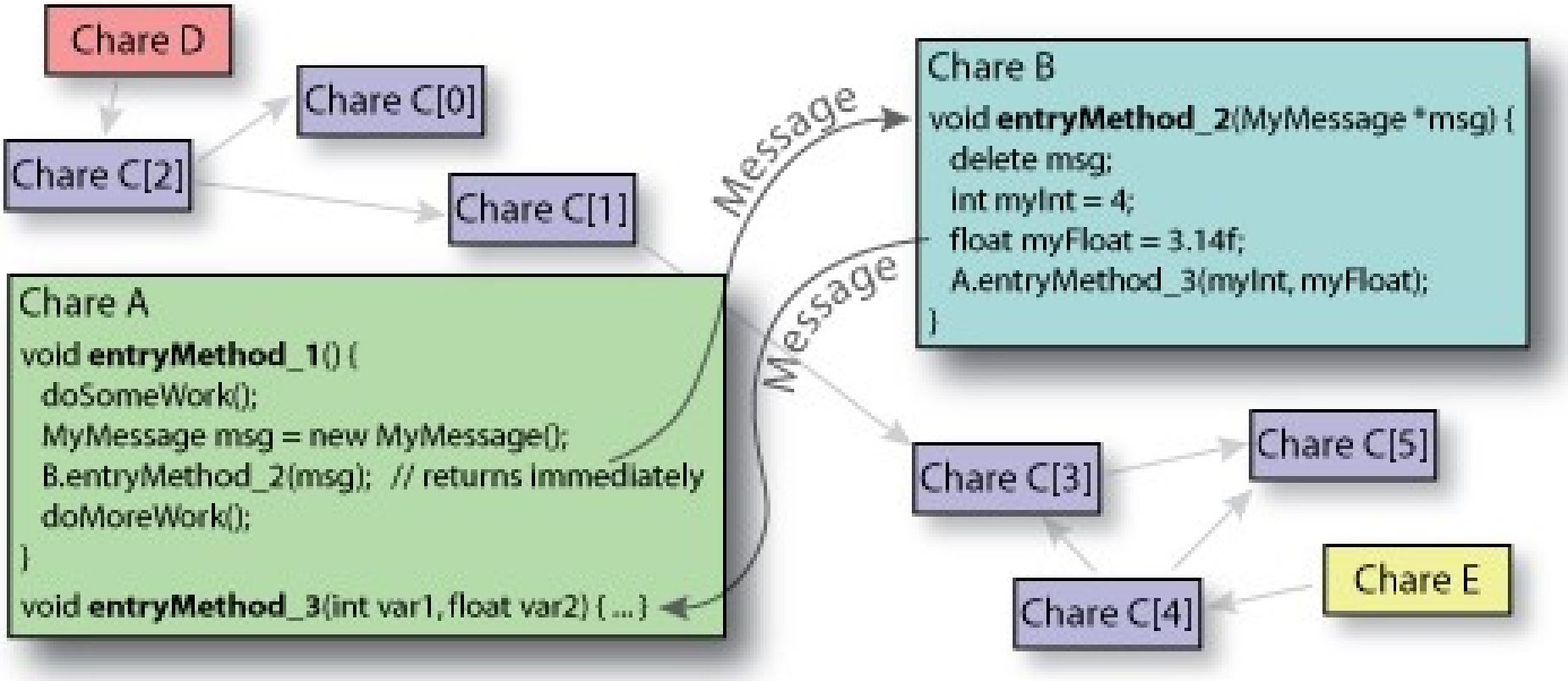
В «чистом» виде имеет много недостатков:

- Очень медленно выполняются последовательные участки. Много времени тратится на поиск нужного узла вычислений.
- Большое количество пересылок.



DataFlow

Поэтому используются гибридные варианты.



Runtime Systems -

вычислительное окружение, доступное во время выполнения компьютерной программы. В среде выполнения, как правило, невозможно изменение исходного текста программы, но может наличествовать доступ к переменным окружения операционной системы, таблицам объектов и модулей разделяемых библиотек.

Главные задачи Runtime в dataflow языках:

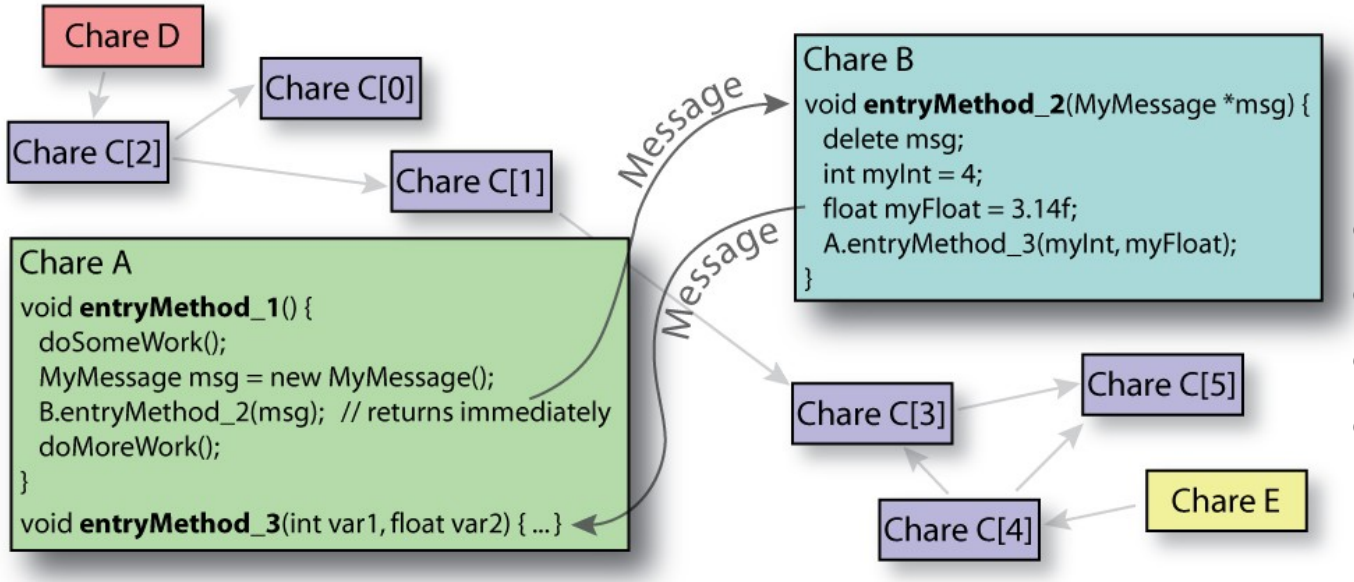
- Распределение графа вычислений на выделенные процессоры.
- Перестройка графа программы в случае необходимости.
- Отказоустойчивость и надёжность.

Charm++ Runtime System

Задачи:

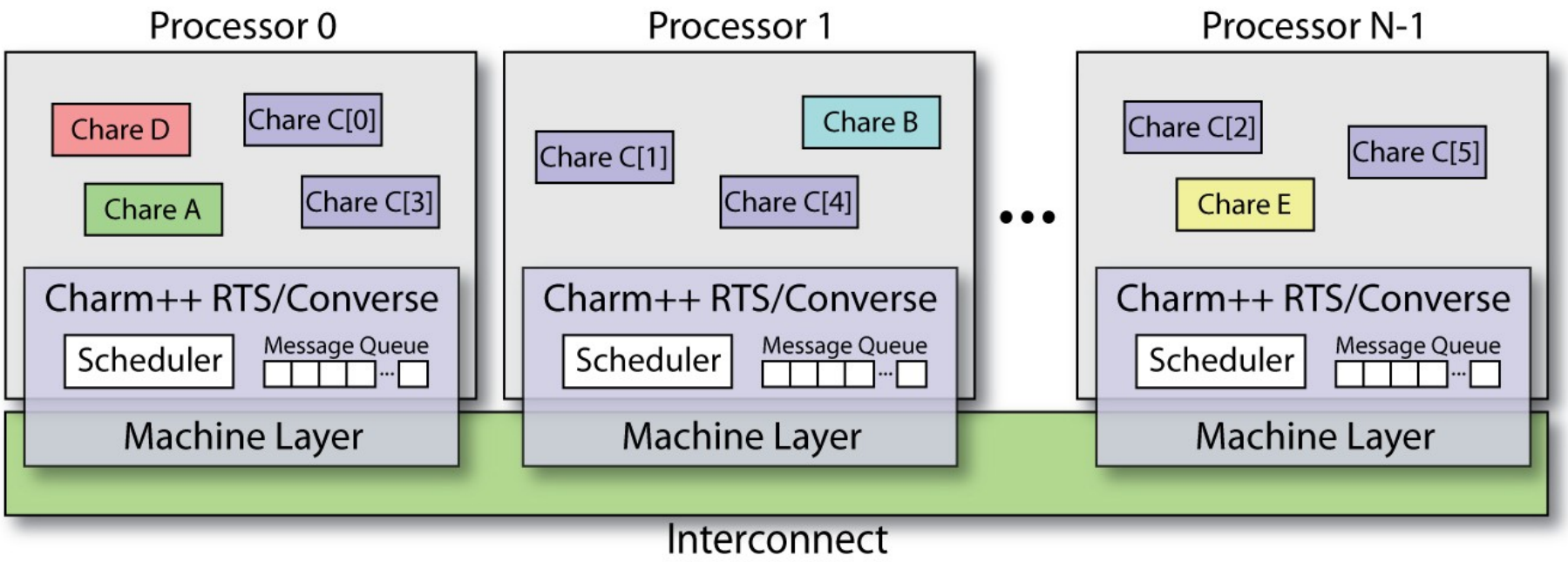
- Отображение chare объектов на физические процессоры
- Балансировка нагрузки chare объектов
- Маршрутизация сообщений
- Создание контрольных точек
- Отказоустойчивость
- Динамическое перераспределение материальных ресурсов

Отображение chrare объектов на физические процессоры



Компоненты Charm++ RunTime System

- Машинный уровень
- Converse надстройка
- Очередь сообщений
- Планировщик



Машинный уровень

Нижний слой Charm++ RTS. Машинный слой содержит определённый системный код, благодаря которому более высокие уровни могут выполнять базовые задачи, такие как пересылка сообщений и др. Когда Charm++ RTS собрана, пользователь должен указать какой машинный слой должен быть использован.

One of the Machine Layers: Cluster of Linux Workstations, IBM's Blue Gene\L, SGI's Altix, Cray's XT3, Infiniband, Myrinet, Ethernet, and more

Converse

Слой над машинным слоем. Данный слой абстрагирует другие слои высокого уровня от возможностей машинного слоя. И предоставляет функционал, не поддерживаемый в конкретном машинном слое.
К примеру, Broadcast может быть не реализован в данном машинном слое. Тогда, при его вызове запуститься реализация, предоставляемая уровнем Converse.

Очередь сообщений

Очередь сообщений действует как пункт сбора сообщений, поступающих к определённому элементу обработки. Очередь делится для всех chares, находящихся на данном элементе обработке. Для базовых сообщений, очередь сообщений работает как обычная очередь(входящее сообщение записывается в конце очереди, обрабатываются в порядке поступления и тд)

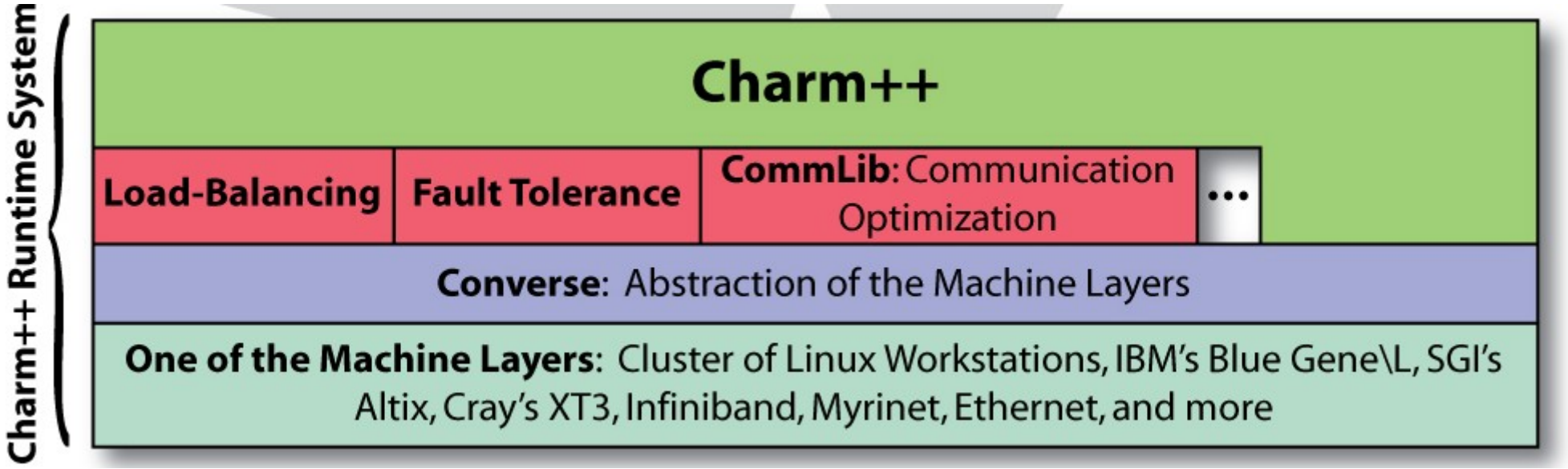
Есть и другие типы сообщений, которые поддерживаются Charm + +: expidited messages, прямые собщения, Сообщения с преоритетом, определённым пользователем. При использовании данных сообщений очередь уже работает не как обычная очередь.

Планировщик

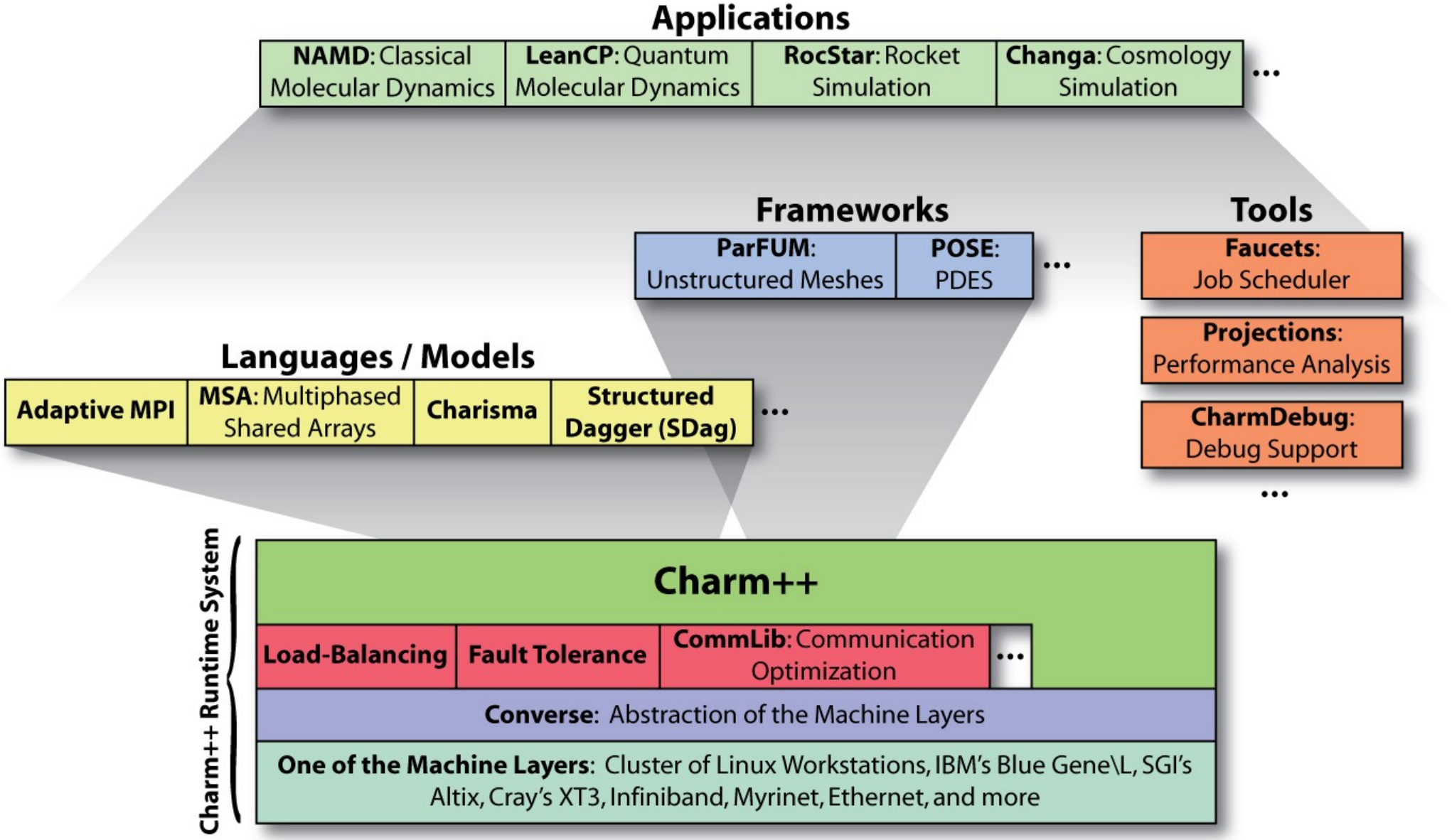
Планировщик является компонентом уровня Converse который заботится о выборе сообщение из очереди сообщений и выполнения их. Запись сообщения в очереди сообщений представляет собой триплет (данные сообщения, Chare объект и метод ввода для выполнения). После того, как планировщик выбрал сообщение для выполнения, он начинает выполнять связаный метод ввода. Этот метод ввода продолжает выполняться, пока либо он не завершит, либо пока управление не будет возвращено обратно Charm++ Runtime System. Из-за этого гарантируется, что только одна запись способна выполнятся в любой момент времени на определенном элементе обработки. Одним из преимуществ такого подхода является защита переменных пользователя, доступных несколькими методами ввода.

Дополнительные модули

Естественно в Charm++ RTS присуствуют модули, обеспечивающие балансировку нагрузок, отказоустойчивость, оптимизации связей, прогнозов и др.



Charm++ RTS поддерживает другие языки и модели...



StarPU

StarPu — программный инструмент, позволяющий использовать вычислительные мощность CPU и GPU, при этом освобождающий от необходимости адаптировать программы к конкретным машинам или вычислительным блокам.
В основе StarPu лежит runtime библиотека, которая отвечает за планирование выполнения задач на разнородных GPU/CPU машинах.
StarPU поставляется в виде расширения языка C, а также в виде внешнего интерфейса OpenCL.
Основная структура данных:

codelet — описание вычислительного ядра.

Task — задача. Выполнение задачи — применение определенных наборов данных на codelet.

Зависимости выстраиваются благодаря callBack и приоритетам.

StarPU

В «библиотеке» StarPu большое внимание уделено расписанием выполнения задач. Есть возможность указывать приоритеты задач(starpu_task::priority). Выбрать одного из планировщиков(STARPU_SCHED) или даже написать его самому. Планировщики:

- **eager** Планировщик использует центральную очередь задач, из которой «рабочий» берёт задачи. Если задача не с нулевым приоритетом, то она попадает в начало очереди.
- **prio** Планировщик использует центральную очередь, но сортирует задачи по приоритетам(от -5 до 5)
- **random** Планировщик распределяет рандомно задачи по всем вычислительным мощностям
- **ws (work stealing)** Планировщик распределяет задачи между «работниками» по умолчанию. Когда работник освобождается, он «ворует» задачи у самого невнимательного/занятого работника
- **dm (deque model)** Планировщик использует модели производительности выполнения задач для использования Heft симуляционной стратегии
- **dmba**(deque model data aware) Дополнительно используется статистика по передаче данных

StarPU
Колибровки.

Есть возможность колибровки времени обработки данных на структуре codelet. StarPu автоматически колибрует не колиброванные codelet.
Есть возможность колибровки в реальном времени и игнорировании колибровок, в случае не статичной работы программы.
Есть возможность явно указать производительность системы, без выполнения программы, на основе прошлых запусков.
Есть возможность колибровки времени передачи сообщений.

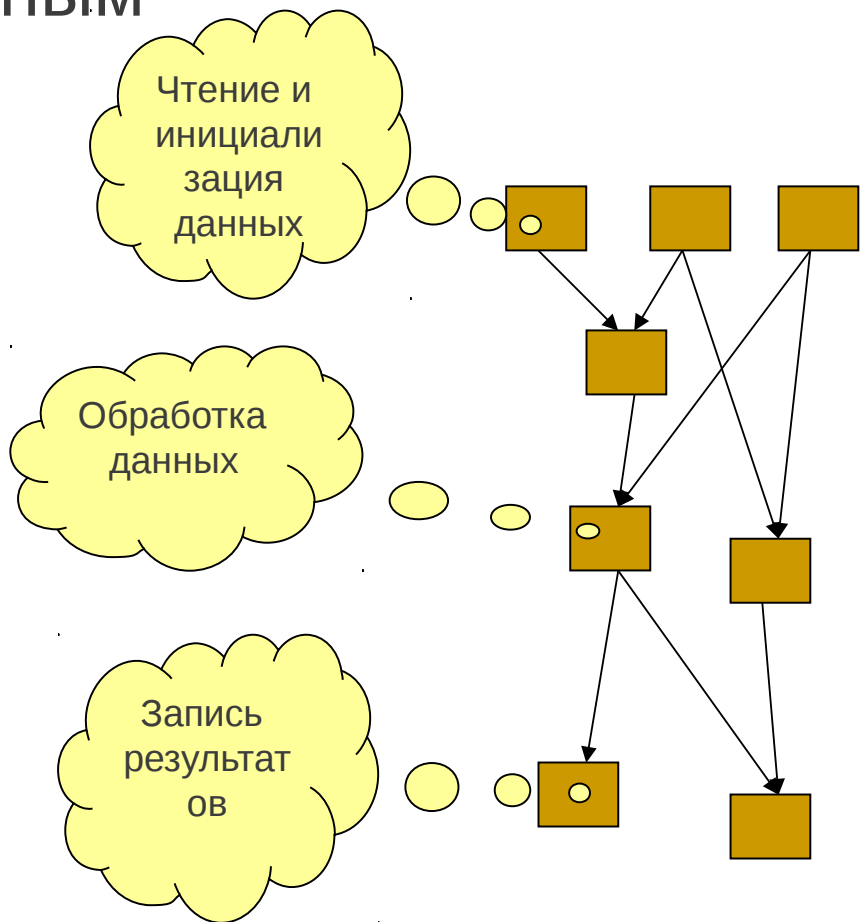
Можно также производить разнообразные симуляций моделей разных codelet на домашнем компьютере.

“Parus”

- Алгоритм решаемой задачи представляется как ориентированный граф:
  - вершины - вычисления(C++)
  - рёбра - зависимость по данным

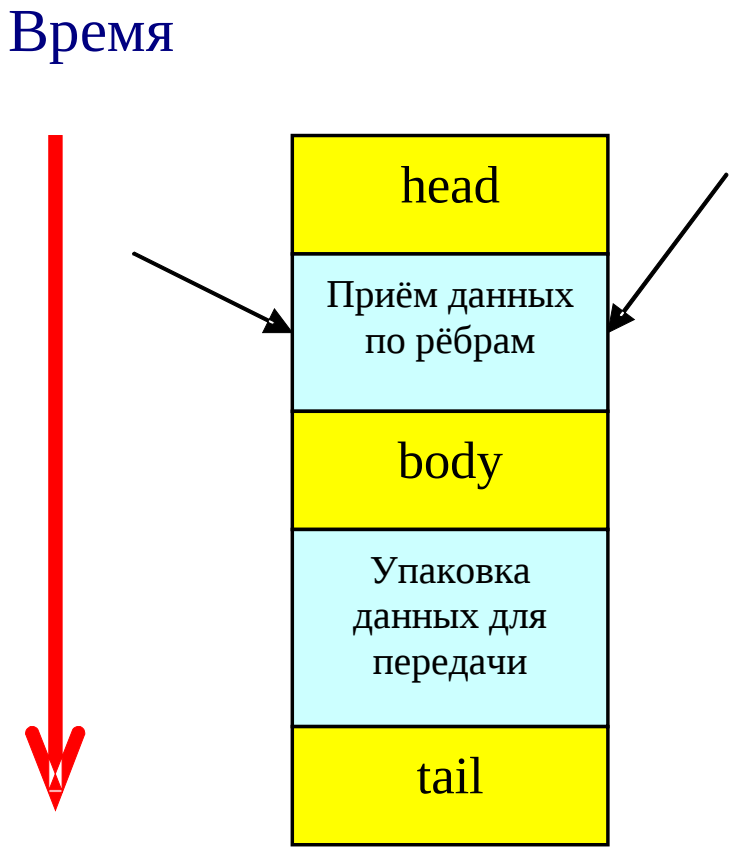
Программа описывается как сеть из

- вершин – истоков
- внутренних вершин
- вершин – стоков



Структура вершины графа

- head – код содержит описания переменных и начальную инициализацию данных необходимых узлу.
- body – код выполняемый после получения всех необходимых данных.
- tail – код содержит действия по подчистке памяти и утилизации данных.



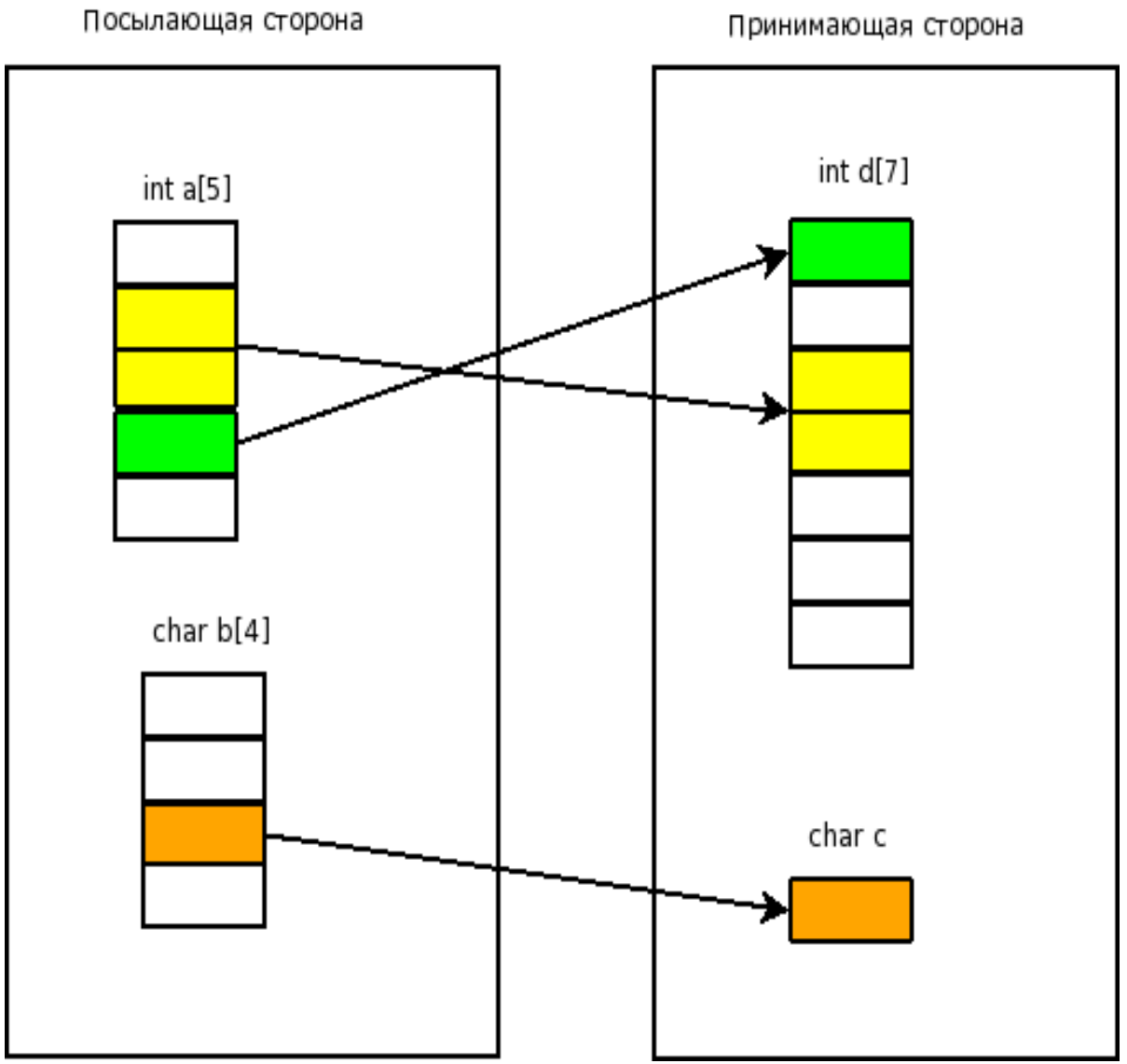
Пример вершины

<NODE_BEGIN>
number 1
type 1
weight 100
layer 2
num_input_edges 1
edges (1)
num_output_edges 1
edges (2)
head "head"
body "node"
tail ""
<NODE_END>

Пример ребра

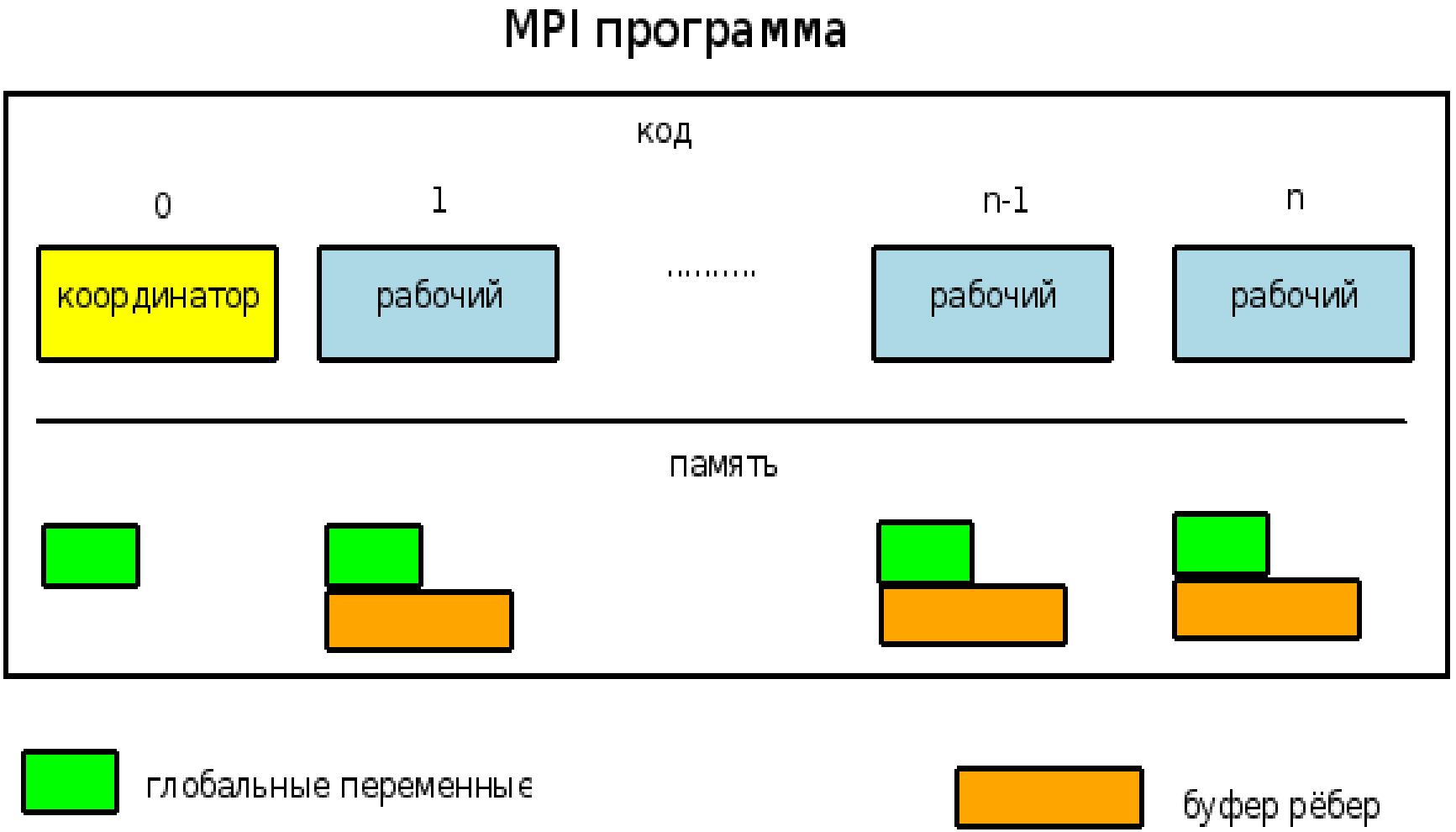
```
<EDGE_BEGIN>
  number 2
  weight 2
  type   GRAPH_NONE
  num_var 1
  num_send_nodes 1
  send_nodes ( 2 )
  num_recv_nodes 1
  recv_nodes ( 5 )
  <SEND_BEGIN>
    <CHUNK_BEGIN>
      name "*data_out"
      type GRAPH_DOUBLE
      left_offset "0"
      right_offset "window_size+F_LEN"
    <CHUNK_END>
  <SEND_END>
  <RECIEVE_BEGIN>

    <CHUNK_BEGIN>
      name "*data_out"
      type GRAPH_DOUBLE
      left_offset "0"
      right_offset "window_size"
    <CHUNK_END>
  <RECIEVE_END>
<EDGE_END>
```

Структура ребра графа



- Данные разбиты на кусочки (чанки), каждый отвечает за свой фрагмент массива.
- Фрагменты принимающей стороной могут быть собраны в другом порядке

Схема runtime системы PARUS-программы



Режимы назначения вершин на обработку MPI-процессам “PARUS”

- Статический
  - Назначение вершин происходит согласно составленному заранее расписанию
- Динамический
  - Назначение вершин происходит согласно текущей ситуации
- Комбинированный
  - Предоставляется возможность учитывать пожелания разработчика граф-программы в спорных ситуациях по назначению вершины на MPI-процесс

Спасибо за внимание.

Для дополнительной информации советую посетить:

Charm++:

- http://charmplusplus.org/ Раздел Learn

StarPu:

- http://runtime.bordeaux.inria.fr/StarPU/ Особое внимание на документацию.

PARUS/Frigate:

- Сальникова Алексея
- http://frigate.sourceforge.net/
- http://sourceforge.net/apps/mediawiki/parus/