# Среда параллельного программирования Parallel Programming Kit

*П. А. Васёв, ИММ УрО РАН, г. Екатеринбуг*

Ведётся разработка среды параллельного программирования. Среда основана на известной модели "издатель-подписчик" и является её своеобразной реализацией. Существует несколько успешных сред, работающих по этой же модели, например:

- ADIOS2 (The Adaptable Input/Output System version 2);
- ROS2 (Robot Operating System, применяется в робототехнике);
- Apache Pulsar и Kafka.

Представляемая среда Parallel Programming Kit интересна тем, что позволяет реализовать разнообразные сценарии, такие как:

- модель графа задач (task-graph, asynchronous many-task systems, см [1-5]);
- метод итераций [6] — вариация на тему оптимизации выполнения графа задач;
- метод схем для графов задач [7];

К разным прикладным задачам удобно применять разные сценарии и подходы. Например, иногда удобно формировать граф задач. В другой ситуации граф задач напрямую оказывается неэффективен, и удобнее использовать метод схем. Среда позволяет решать разнообразные задачи вычислений, и в частности задачи онлайн-визуализации, с помощью разных подходов.

Основные понятия среды:

**Вычислительная среда** — совокупность компьютеров, соединённых сетью, и процессы, выполняющиеся на них. Процессы имеются ввиду как в смысле процессов операционной системы (ОС), так и в логическом смысле (процессы Хоара).

**Сообщение** — передаваемая единица информации. С сообщением связано значение, которое без ограничения общности будем считать двоичным блоком данных.

**Канал** — логический процесс передачи информации. По каналу передаются сообщения.

**Пространство имён каналов** — общее для вычислительной среды пространство имён, в котором именам сопоставлены каналы. Эти имена будем называть идентификаторами каналов.

Основные операции среды:

**Открыть канал на запись**

$$\text{open} : \text{id} \rightarrow \text{channel}$$

где id – идентификатор канала, channel — локальный для программы объект для взаимодействия с каналом.

**Послать сообщение в канал**

$$\text{put}: \text{channel}, \text{value} \rightarrow \text{nil}$$

где channel — локальный объект для взаимодействия с каналом; value — значение для передачи в сообщении.

**Получать сообщения канала**

react: id, fn → rhandle

где id – идентификатор канала; fn — локальная функция, которой будет передаваться управление при поступлении сообщений в канале, с аргументом — значением сообщения; rhandle – локальный объект, позволяющий остановить получение сообщений. Таким образом процесс, вызвавший операцию react, становится "получателем" сообщений заданного канала.

Необходимо отметить, что технически отправители и получатели сообщений того или иного канала могут находиться на разных машинах. Нет ограничений на то, сколько пишущих или читающих сторон у канала.

**Добавить связь между каналами**

bind: source_id, target_id → lhandle

где source_id – идентификатор канала-источника; target_id – идентификатор канала-приёмника; lhandle – локальный объект, позволяющий удалить связь. Запускает процесс копирования сообщений из канала-источника в канал-приёмник. Связь между каналами, как понятие среды, ортогональна понятию канала.

Важные особенности реализации среды:

В ходе экспериментов выяснилось, что для эффективной работы необходима реализация следующих особенностей.

1. **Прямая пересылка сообщений.** Передача сообщений между отправителями и получателями происходит напрямую, без посредников (брокеров и т. п.). Для этого организуется ведение общего реестра (кто на что подписан) и передача участков этого реестра отправителям. Отправитель имеет полный список получателей сообщений и высылает сообщения им напрямую. При изменении реестра – отправителям рассылаются соответствующие обновления. В момент отправки сообщений дополнительных запросов к реестру не требуется.

2. **Передача сообщений по связям сразу на отправителе**. Канал может быть источником в тех или иных связях (см. выше). При выполнении операции отправки сообщения в канал связи обрабатываются тут же, на месте. Отправитель имеет полный список каналов-получателей (из реестра, см. выше) и вызывает операцию записи сообщения в эти каналы. Это позволяет настраивать сложную маршрутизацию сообщений между каналами. При этом сообщения в итоге могут фактически и не покидать процесс ОС отправителя (см. далее).

3. **Передача ссылок вместо данных**. В случае, если отправитель и получатель находятся в одном процессе ОС, то значения сообщений передаются по ссылке в оперативной памяти. При этом данные не копируются, и никуда не перемещаются. Эта особенность оказалась важной потому, что оказалось удобно и эффективно запускать разные логические процессы и "расселять" их части на одни и те же процессы ОС. В этом случае такие процессы взаимодействуют посредством передачи сообщений в каналах с той же скоростью, как если бы находились в коде одной программы и вызывали функции друг друга напрямую.

4. **Передача сложных объектов**. Некоторые сущности, которые требуется передавать в сообщениях, имеют сложную структуру в оперативной памяти (например, некоторые реализации деревьев). Однако для передачи этой сущности в сообщении, в ситуации как в п. 3 выше, А) нет необходимости записывать её в двоичный формат: сущность можно передать по ссылке (обеспечив соответствующий контроль передачи владения и т. п.). При этом иногда всё же необходимо Б) передавать такие сущности и по сети. Заранее неизвестно, какой случай наступит, А или Б. Поэтому предлагается оснащать передаваемые сущности операциями сериализации. В случае А сущность передаётся по ссылке, а в случае Б применяется сопоставленная сущности операция сериализации сущности в двоичный поток.

Реализация этих особенностей позволила существенно повысить производительность среды. Другими перспективными особенностями автор считает:

1. Изучить вопрос целесообразности пакетной передачи сообщений. Например, в среде ADIOS2 поддерживается такой сценарий, когда пользователь сначала записывает набор сообщений (операции put), а затем вызывает операцию "а теперь передаём" (операция PerformPuts).

2. Возможности оптимизации передачи сообщений. Например, несколько получателей канала могут находиться в одном сегменте сети, существенно удалённом по отношению к отправителю. Возможно, что сообщение эффективнее один раз передать в этот сегмент, и уже на месте "раздать" сообщение получателям. Таким образом речь идёт о введении брокеров, но более тонких, по ситуации и необходимости.

3. Подключение высокоэффективных сетевых протоколов. Сейчас для сетевой передачи применяется протокол TCP, но в суперкомпьютерах применяются и более эффективные протоколы, например семейства RDMA. Для этого рассматривается фреймворк UCX.

Среда применяется в разных экспериментах и в рабочих задачах, например [8]. Исходные коды среды публикуются по адресу https://github.com/pavelvasev/ppk.

Автор благодарит коллег за поддержку и обсуждение вопросов. Особенную благодарность автор выражает М. О. Бахтереву (ИММ УрО РАН).

## Литература

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. - СПб.: БХВ-Петербург, 2002. - 608 с.
   Прим: *граф алгоритма*, стр. 191.
2. Прихожий, А.А. Распределенная и параллельная обработка данных // Белорусский национальный технический университет, Кафедра "Программное обеспечение вычислительной техники и автоматизированных систем". - Минск: БНТУ, 2016. - 91 с.
   Прим.: *граф задач*, стр. 44.
3. Sally Stevenson, Stephen Jones and Fred Oh, Enabling Dynamic Control Flow in CUDA Graphs with Device Graph Launch // 12 dec 2022, Nvidia developer blog.
4. Reazul Hoque, Thomas Herault, George Bosilca, and Jack Dongarra. 2017. Dynamic task discovery in PaRSEC: a data-flow task-based runtime. In Proceedings of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA '17).

Association for Computing Machinery, New York, NY, USA, Article 6, 1–8. https://doi.org/10.1145/3148226.3148233

5. Pavel Vasev, A Computational Model for Interactive Visualization of High-Performance Computations // In: Voevodin, V., Sobolev, S., Yakobovskiy, M., Shagaliev, R. (eds) Supercomputing. RuSCDays 2023. Lecture Notes in Computer Science, vol 14389. Springer, Cham. https://doi.org/10.1007/978-3-031-49435-2_9
6. Васёв П.А., Метод итераций для графов задач // Параллельные вычислительные технологии – XVIII всероссийская конференция с международным участием, ПаВТ'2024, г. Челябинск, 2–4 апреля 2024 г. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2024. С. 182. ISBN 978-5-696-05450-6. DOI: https://doi.org/10.14529/pct2024
7. П. А. Васёв, Один подход к онлайн-визуализации параллельных вычислений // Тезисы XIX Международной конференции СУПЕРВЫЧИСЛЕНИЯ И МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ, 20 – 24 мая 2024 г., г. Саров, С. 100. URL: https://www.cv.imm.uran.ru/e/3241938
8. Vasev P.A., 3D Visualization of Asynchronous Many-Task Scheduling Algorithm // Scientific Visualization journal. 2023. 15.4: 92 - 111, DOI: https://doi.org/10.26583/sv.15.4.08