Техническая документация «DistComp»

Автор: Агеев Данил Евгеньевич

Содержание:

Предисловие, доп. информация.

DistComp – это программа, позволяющая распределять вычисления по алгоритму гипотезы Коллатца между несколькими компьютерами.

Далее в тексте будут использованы наименования сторон программы – *сервер* и *клиент*. Сервер распределяет вычисления между клиентами, а клиенты выполняют эти вычисления. При первом запуске Клиент генерирует UUID (идентификатор) и сохраняет его в файл `uuid.txt`, идентификатор клиента служит для определения и запоминания конкретного клиента сервером. Если клиент прекратил работу, то в любой момент сможет продолжить ее, подключившись к серверу с таким же идентификатором, который был в прошлый раз.

Ведется логирование как в файл, так и в стандартный вывод (`STDOUT`), находящийся в папке «log». Для каждой сессии создается новый файл, названный по времени его создания в формате «год-месяц-день-час-минута-секунда.txt».

Распределение вычислений.

Распределение вычислений происходит по системе выделения интервалов для каждого из клиентов. Всего в программе предусмотрено два типа интервала – *выделенный* и *свободный*. Выделенный интервал – это интервал [a…b], содержащий целые числа, над которым работает клиент. В нем задаются два параметра – «от числа N» (включительно) и до «до числа N» (исключительно). Так например, интервал, обозначенный [5…8] будет покрывать числа 5, 6 и 7. Свободный же интервал содержит все целые числа, которые не были выделены, и обозначается он как [a...inf].

Давайте рассмотрим пошаговую обработку интервалов сервером. Зеленым цветом обозначим свободный интервал.

Допустим точкой отсчета[1] у сервера будет 3, а буфер клиента[1] – 4 (сколько чисел выделяется за раз) Значит свободный интервал здесь [3…inf] Сейчас сервер инициализирован и готов к распределению работы.

Через некоторое время к серверу подключились два клиента (в хронологическом порядке: красный, синий). Сервер выделяет им по интервалу. Красному – [3…7], а синему – [7…11] (т.к. буфер клиента равен 4, каждый получает по 4 числа) А свободный интервал стал [11…inf]

Клиенты начали работать, и спустя некоторое время красный клиент закончил свою работу, в то время как синий сделал всего лишь часть от своей. Красный клиент сообщает серверу о том, что он закончил работу на своем выделенном интервале, а сервер выделяет ему еще – [11…15], при этом свободный интервал передвигается до [15…inf]

При потере соединения с клиентом, сервер запоминает его интервал и место, на котором он остановил вычисление, и сохраняет его до того момента, когда этот же клиент подключится.
Примечание: Интервал, однажды выделенный одному клиенту, никогда не станет частью свободного.

[1] Значение точки отсчета и буфера клиента задаются при инициализации сервера и являются положительными целыми числами.

Потоки, работающие на стороне сервера.

1. «SocketAcceptor» – принимает, клиентов, пытающихся подключиться к серверу. `ServerSocket.accept()` блокирует поток, поэтому для принятия клиентов необходимо выделить отдельный поток.
2. «Server-Console» – читает и обрабатывает запросы из потока стандратного ввода (`STDIN`). Подробнее о доступных командах в разделе «Запуск настойка программы»
3. «ClientHandler» – для каждого подключенного клиента запущен свой `ClientHandler` поток (различаются они по `SocketAddress` и UUID, приписанному в конец названия потока). Здесь происходит чтение и обработка информации из потока ввода сокета клиента (`java.lang.Socket`)

Потоки, работающие на стороне клиента.

1. «Client-ServerCom» – читает и обрабатывает запросы из потока ввода сокета сервера (`java.lang.Socket`)
2. «Syracuse-Solver» – выполняет вычисления на заданном интервал

Запуск и настройка программы.

Серверная сторона запускается с единственный обязательным аргументом – портом, на котором будет работать сервер.
Пример команды запуска программы как сервер в Windows:

```
java -jar distcomp.jar server 12345
```

У стороны клиента же два аргумента – порт и адрес сервера.
Пример:

```
java -jar distcomp.jar client 12345 127.0.0.1
```

Дополнительно, добавив `–enableDebugLogging` в конец аргументов запуска, можно включить логирование на уровне отладки. Это значительно замедлит вычисление, т.к. каждое обработанное число нужно будет записать в файл.

Для инициализации вычисления необходимо написать следующую команду в стандартный ввод сервера:

```
init [1] [2] [3] [4]
```

Разберем аргументы:

1. Число, с которого начнется вычисление. Если значение этого аргумента равно a, то свободный интервал будет установлен на [a…inf]
2. Буфер клиента, определяющий сколько чисел сервер будет выделять каждому клиенту за раз.
3. Время в миллисекундах (N). Если одно число занимает больше N миллисекунд на обработку, то каждые N миллисекунд серверу будет отправлен отчет об этом. Отчеты записываются в файл журнал вычислений[2] как строки в формате «X ms >> Y», где X – время в миллисекундах, сколько заняла обработка одного числа, кратно N; Y – само число.

4. Время в миллисекундах (L). Каждые L миллисекунд будет производиться запись в журнал вычислений[2] соответствующая интервалу, который был вычислен конкретным клиентом за последние L миллисекунд. Каждый раз, когда клиент заканчивает работу на предварительно выделенном ему интервале, этот интервал записывается в журнал, и таймер авто-записи сбрасывается, т.е. если клиент успевает обработать выделенный интервал быстрее, чем за L миллисекунд, авто-запись не произойдет.

Для корректного выключения сервера используйте команду:

`x`

Примечание: При аварийном выключении сервера информация о прогрессе вычисления клиентов не сохранится и единственным способом восстановления будет –вручную переписать файлы сохранений через журналы вычисления[2].

[2] Журнал вычислений – файл с именем `COMPLOG_%.txt`, где `%` – UUID клиента. Создается отдельный журнал для каждого клиента. В него записываются интервалы, которые клиент успешно вычислил, а также отчеты о том, что какое-то число занимает слишком много времени для обработки.

Эта версия документации была написана 25 мая 2016 г. для версии ПО 2.0-a3 и не является последней.