BioinfoWF – веб-сервисы и пакет конвейерной обработки для решения задач биоинформатики*

© М.А. Генаев[1], К.В. Гунбин[1], Д.А. Афонников[1,2]

[1]Институт цитологии и генетики СО РАН, г. Новосибирск

[2]Новосибирский государственный университет

mag@bionet.nsc.ru

Аннотация

Рассматривается система конвейерной обработки биологических данных для решения задач биоинформатики. Элементами конвейера являются вычислительные модули и связи между ними. Разработаны модель описания вычислительных модулей и схемы конвейеров на языке XML. Система реализована как клиент-серверное приложение: клиентская часть реализована в виде веб-сервиса; серверная часть позволяет запускать отдельные элементы конвейера на высокопроизводительном кластере, что дает возможность масштабировать расчеты. На основе системы разработан ряд готовых конвейеров, в том числе конвейеры для анализа моделей молекулярной эволюции генов и белков SAMEM [1].

1 Введение

В биологии накапливается огромное число данных. Для их анализа методами биоинформатики часто необходима последовательная компьютерная обработка в автоматическом режиме, которая может быть реализована в виде конвейера. Например, при построении филогенетического дерева для нескольких последовательностей белков требуется последовательное решение ряда таких задач, как:

- поиск гомологов, т. е. белков, выполняющих схожие функции или имеющих сходные последовательность и структуру;
- выравнивание последовательностей;
- фильтрация получившейся выборки;
- построение филогенетического дерева.

Процесс обработки усложняется разнородностью форматов входных и выходных данных. Кроме того, каждый этап обработки может быть реализован разными вычислительными программами и алгоритмами, в зависимости от свойств анализируемых данных.

Для решения подобного сорта задач в разных областях науки создаются системы конвейерной обработки данных. К ним относятся, например, DiscoveryNet в молекулярной биологии [2], SEEK в экологии [3], GriPhyn в физике элементарных частиц [4]. В области биоинформатики наиболее известны следующие системы конвейерной обработки.

Taverna project [5] – графическая среда для управления и запуска конвейеров, реализованная на языке Java. Главная идея проекта – интеграция различных веб-сервисов и конструирование из них конвейеров. Для формального описания веб-сервисов и структуры конвейера был разработан специальный язык XScufl (XML Simple conceptual unified flow language).

Biopipe [6] – система, которая, напротив, предоставляет множество готовых шаблонов для популярных биоинформатических программ и баз данных. Не поддерживает асинхронное выполнение конвейеров, предполагая только последовательную обработку данных.

В настоящей работе для решения ряда специфических задач в области эволюционной биоинформатики мы разработали систему, которая позволяет конструировать конвейеры биоинформатической обработки данных и выполнять их. Пользователю предлагается работать с уже готовыми схемами конвейеров через веб-интерфейс. Узлы в конвейере – это вычислительные модули, которые запускаются на счёт или непосредственно на сервере, или на вычислительном кластере с использованием Sun Grid Engine.

С помощью данной системы нами был реализован ряд конвейеров для анализа молекулярной эволюции последовательностей ДНК и белков.

2 Архитектура системы

Система BioinfoWF – клиент-серверное приложение (рис. 1), которое решает задачу конвейерной обработки данных.

Конвейер – это набор вычислительных модулей, которые представляют собой программы, запускаемые в консольном режиме (в среде Linux). Управляющие параметры (названия входных и выходных файлов, параметры алгоритмов) передаются в рас-

Труды 12[й] Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» – RCDL'2010, Казань, Россия, 2010

четные модули через командную строку или переменные окружения. В ходе выполнения задачи выходные данные одного модуля могут подаваться на вход другому модулю.

Рис. 1. Основные структурные элементы системы BioinfoWF

Разработанная нами система позволяет интегрировать любые вычислительные модули, организованные подобным образом, при условии заданного порядка их выполнения. При этом данные могут находиться как на локальной машине, так и на удалённой. Схема интеграции (порядок выполнения процедур) описывается на языке XML. Имея готовые схемы, пользователь может запускать конвейер как консольное приложение или с использованием веб-интерфейса, который генерируется автоматически. В последнем случае пользователь может управлять поведением конвейера, изменяя его схему и входные данные для вычислительных модулей. Для визуализации и редактирования входных и выходных данных имеются возможности подключения внешних программ, реализованных, как правило, в виде Java Applet приложений. Серверная часть, реализованная на языке Perl, выполняет запуск конвейера и отлеживает статус выполнения каждого вычислительного модуля. Также она предоставляет возможности запуска ресурсоёмких узлов конвейера на вычислительном кластере и поддерживает функцию параллельного запуска вычислительных модулей в случае, если это позволяет топология конвейера.

Способ выполнения задачи (на локальной машине или удаленном кластере) указывается в описании вычислительного модуля. Для передачи данных на кластер монтируется сетевая файловая система sshfs. При работе с кластером для запуска и отслеживание статуса выполнения заданий менеджер задач BioinfoWF использует программное обеспечение, установленное на кластере. Текущая версия BioinfoWF поддерживает такие системы управления задачами, как Sun Grid Engine и Altair PBS Pro. Удалённый запуск команд на кластер реализован так же с помощью протокола ssh.

3. Язык описания конвейера

Для формального описания схемы конвейера и вычислительных модулей, из которых он состоит, нами был разработан язык на основе формата XML. Описание конвейера состоит из двух файлов (рис. 2).

Первый описывает вычислительные модули, второй задаёт топологию конвейера.

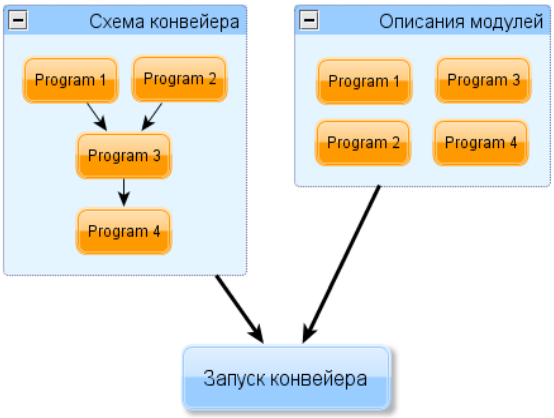

Рис. 2. Описание любого конвейера включает описание вычислительных модулей, вовлеченных в конвейер и связи между этими модулями

Описание вычислительных модулей состоит из следующих разделов:

- *Входные файлы* – описывает, какие входные файлы подаются на вход модулю; для каждого файла указываются его идентификатор, описание, формат файла;
- *Выходные файлы* – описывает, какие файлы возвращает вычислительный модуль;
- *Параметры и опции* – содержит описание параметров и опций для вычислительного модуля. Для каждого параметра задаются идентификатор, описание, тип параметра (например, строка, число или бинарное значение), значение по умолчанию, внешний вид поля запроса значения для параметра на странице веб-браузера;
- *Правила генерации командной строки* – программа на языке Perl, которая генерирует командную строку для запуска вычислительного модуля. На вход ей подаются две хеш-таблицы: первая – со списком входных и выходных файлов, вторая – с опциями запуска исполняемых модулей. Ключами в таблицах являются, соответственно, идентификаторы файлов и опций. С использованием этих данных программа формирует строковую переменную $cmd, которая и будет являться командной строкой запуска вычислительного модуля;
- *Правила поведения пользовательского интерфейса* – опциональная секция. Это программа на JavaScript, которая обрабатывает действия пользователя и в зависимости от этих действий динамически меняет веб-интерфейс модуля.

Рассмотрим пример описания в формате XML команды kill, которая прекращает выполнение какого-либо вычислительного процесса. Программа на

вход принимает идентификационный номер процесса, который надо завершить.

```
1  <programs>
2  <program name="Kill" exe="kill">
3  <description>
4    kill - terminate a process
5  </description>
6
7  <output>
8  <file id="stdout" type="text"
9  name="STDOUT"
10 description="Standard output" />
11 <file id="stderr" type="text"
12 name="STDERR"
13 description="Standard error" />
14 </output>
15
16 <options>
17 <option id="PID" name="PID"
18 description="PID" view="text" type="int“
19 default="" />
20 </options>
21
22 <cmdline>
23   $cmd = " $options{PID} ".
24   "1>\"$files{stdout}\" ".
25   "2>\"$files{stderr}\"";
26 </cmdline>
27
28 </program>
29 </programs>
```

Вторая строчка описывает название вычислительного модуля и путь, где располагается исполняемый файл модуля. В нашем случае kill – это команда окружения bash, поэтому указания полного пути не требуется. Секция output (строки 7 – 14) описывает выходные файлы, в этом примере описываются два файла с идентификаторами *stdout* и *stderr*, которые мы в дальнейшем ассоциируем со стандартными потоками вывода 1 и 2 соответственно. Аналогичным образом описывается секция input, для входных файлов, в нашем примере входных файлов нет, поэтому секция отсутствует. В секции options описывается единственная опция, которая будет передавать id процесса команде kill. Опция имеет тип *int* и представление *text*, которое будет соответствовать *<input type=”text” />* при генерации веб-интерфейса. Значение по умолчанию для опции не задано. Секция cmdline описывает правила генерации командной строки. На входе мы имеем две хеш-таблицы *$options* и *$files*. Ключами в этих хеш-таблицах служат id из секций *input*, *output* и *cmdline*. На выходе необходимо сформировать переменную *$cmd*, которая бы содержала готовую командную строку для вычислительного модуля.

Второй документ описывает в формате XML топологию конвейера. В нём указываются порядок выполнения задач, имена входных/выходных файлов для каждого модуля и значения для каждого параметра или опции при запуске. Для нашего примера с командой kill может быть использован следующий файл:

```
1  <pipeline>
2
3  <node name="Kill_task" parent="undef"
4  program="Kill" status=“undef">
5
6  <file id="stdout"
7  value="/tmp/kill.stdout" />
8  <file id="stderr"
9  value="/tmp/kill.stderr" />
10
11 <option id="PID" value="2745" />
12
13 </node>
14 </pipeline>
```

Строки 3 – 4 определяют вычислительный модуль, который необходимо запустить. Ключ *name* определяет название задачи в конвейере, *program=”kill”* указывает на то, что надо запустить программу kill, которую мы описали выше. Ключ *parent=”undef”* указывает на то, что у этого узла в конвейере нет зависимостей. Если бы был указан родительский процесс, узел не был бы запущен на счет до тех пор, пока не отчитались процессы, его порождающие. Ключ *status=”undef”* определяет начальный статус узла. Статусы каждого узла меняются во время выполнения конвейера, каждый узел может принимать следующие статусы: *undef* (не определен), *started* (запущен на счет), *ended* (завершен), *failed* (завершен с ошибкой). Далее в файле следует определение значений для всех входных/выходных файлов, параметров и опций вычислительного модуля. Строки 6 – 9 определяют имена выходных файлов, а строка 11 – идентификационный номер процесса, который требуется завершить.

4. Серверная часть

Серверная часть отвечает за запуск и выполнение конвейера. Она реализована в виде приложения на языке Perl. На вход приложению подаётся описание схемы конвейера и вычислительных модулей. Приложение запускает конвейер, создавая файл с отчётом в формате xml. В отчёте указывается статус выполнения каждого узла в конвейере. Серверная часть поддерживает параллельный запуск узлов конвейера, при этом максимальное количество потоков определяется в конфигурационном файле приложения. Реализована возможность удалённого запуска ресурсоёмких расчетных модулей конвейера на вычислительном кластере (с использованием систем Sun Grid Engine или Altair PBS Pro). Режим запуска для каждого узла (локальный или удалённый) задаётся в схеме конвейера.

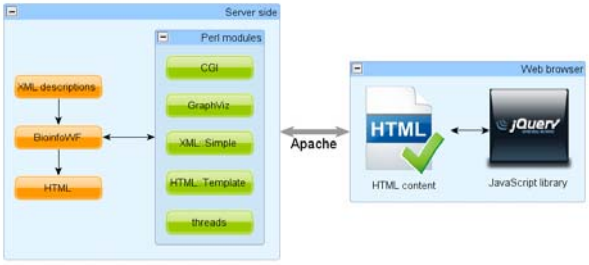

Рис. 3. Структурная схема процесса генерации веб-интерфейса

5. Клиентская часть

Клиентская часть системы реализована в виде веб-приложения. Веб-интерфейс генерируется автоматически на основе описаний вычислительных

модулей и схемы конвейера (рис. 4). Пользователю предлагается работать с уже готовыми схемами. В текущей версии клиентской части возможна работа только с последовательными конвейерами. Однако в ближайшем будущем планируется разработка новой версии веб-интерфейса, который бы позволил конструировать произвольные пользовательские конвейеры любой топологии.

Схема генерации веб-страниц клиентской частью представлена на рис. 3. BioinfoWF получает на вход описание схемы конвейера и вычислительных модулей в формате XML и с использованием библиотеки Perl HTML::Template генерирует файл в формате html. Автоматизация достигается за счет того, что в описании каждого входного параметра указывается тип элемента HTML для его визуального представления (выпадающее меню, радиокнопка, текстовое поле и т. п.; рис. 4).

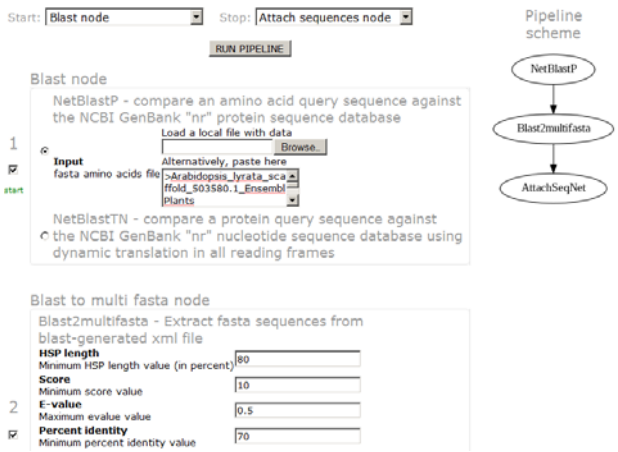

Рис. 4. Веб-интерфейс пользователя системы SAMEM, реализованной на платформе BioinfoWF

Реакция интерфейса на действие пользователя при его работе с HTML-страницами достигается за счет внедрения в описание каждого модуля динамических правил поведения, реализованных с помощью библиотеки jQuery. Динамическое изменение интерфейса удобно использовать, когда существуют зависимости между параметрами исполняемых модулей. Например, в модуле, описывающем задачу множественного выравнивания белковых последовательностей программой Mafft, выбор матрицы сравнения аминокислот обуславливает ряд дополнительных опций, которые зависят от ее типа. При выборе матрицы на веб-странице отображаются только опции, связанные типом выбранной матрицы.

Предложенный подход позволил создавать качественные и эргономичные веб-интерфейсы для любых вычислительных модулей в автоматическом режиме.

В текущей версии интерфейса доступны следующие базовые опции управления конвейером:

- установка входных файлов, параметров и опций для каждого вычислительного модуля в конвейере;
- старт с произвольного узла и остановка на произвольном узле в конвейере;
- отслеживание статуса выполнения каждого вычислительного модуля в конвейере (рис. 5);
- просмотр входных/выходных файлов для каждого этапа расчета в конвейере;
- привязка форматов входных/выходных файлов к различным приложениям для их визуализации.

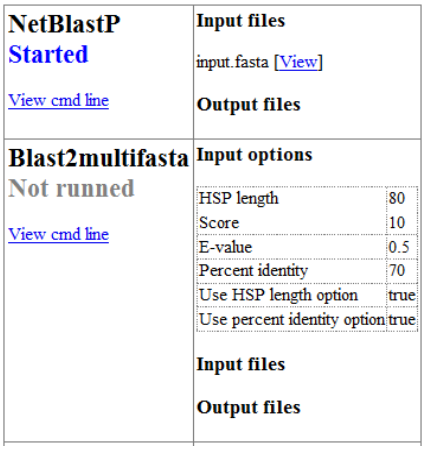

Рис. 5. Веб-интерфейс системы BioinfoWF позволяет отслеживать статус выполнения каждого вычислительного модуля и просматривать входные/выходные данные узлов конвейера

Разработанная нами система BioinfoWF была использована для конструирования и выполнения конвейеров решения задач молекулярной эволюции генов и белков SAMEM [1]. Первый конвейер позволяет последовательно выполнять следующие задачи: (1) множественное выравнивание последовательностей генов; (2) построение филогенетического дерева; (3) реконструкция предковых последовательностей генов во внутренних узлах филогенетического дерева; (4) анализ режима эволюции генов на всех ветвях дерева; (5) сравнение режима эволюции генов с режимом эволюции фенотипических признаков организмов. Второй конвейер выполняет аналогичные процедуры для аминокислотных последовательностей. При этом вычисления в каждом узле конвейера могут быть выполнены одним из нескольких способов. Так, например, для множественного выравнивания пользователю предлагается выбрать один из двух вариантов алгоритмов.

6 Заключение

Предложенный подход позволяет реализовывать системы расчета биоинформатических задач в виде конвейера. Система не требует от пользователя программирования, необходимо только описать параметры вычислительных модулей в определенном формате. Интерфейс конвейера генерируется в автоматическом режиме. С помощью системы реализованы конвейеры по анализу молекулярной эволюции генов и белков.

Литература

[1] SAMEM – Computer System for Analysis of Molecular Evolution Modes, 2010. – http://pixie. bionet.nsc.ru/samem/.

[2] Rowe A., Kalaitzopoulos D., Osmond M., M. Ghanem, Guo Y. The discovery net system for high throughput bioinformatics// Bioinformatics. – 2003. – V. 19. – P. 225-231.

[3] Altintas I., Berkley C., Jaeger E., Jones M., Ludscher B., Mock S. Kepler: towards a grid-enabled system for scientific workflows// Workflow in Grid Systems Workshop in GGF10, Berlin, March 2004.

[4] Deelman E., Blythe J., Gil Y., Kesselman C. Workflow management in GriPhyN. – In J. Nabrzyski, J. Schopf, J. Weglars (Eds). Grid resource management. – Kluwer, 2003.

[5] Oinn T., Addis M., Ferris J., Marvin D., Senger M., Greenwood M., Carver T., Glover K., Pocock MR., Wipat A., Li P. Taverna: a tool for the composition and enactment of bioinformatics workflows. – Bioinformatics. – 2004. – V. 20. – P. 3045-3054.

[6] Hoon S., Ratnapu K., Chia J., Kumarasamy B., Juguang X., Clamp M., Stabenau A., Potter S., Clarke L., Stupka E. Biopipe: a flexible framework for protocol-based bioinformatics analysis. – Genome Res. – 2003. – V. 13, No 8. – P. 1904-1915.

BioinfoWF – web-services and workflow management for bioinformatics analysis

M.A. Genaev, K.V. Gunbin, D.A. Afonnikov

To perform workflow data processing for bioinformatics we developed BioinfoWF system. The BioinfoWF runs under command line on the UNIX-like systems or as a web-service. The workflow or its part can also perform on the multiprocessor cluster systems under Sun Grid Engine.

*Работа выполнена при финансовой поддержке РФФИ (проект 09-04-01641); интеграционных проектов СО РАН №№ 113, 26, 109; Программы РАН «Происхождение и эволюция биосферы»