Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«Самарский государственный аэрокосмический университет

имени академика С.П. Королева

(национальный исследовательский университет)»

Факультет информатики

Кафедра технической кибернетики

Отчет по лабораторной работе № 1 по дисциплине

«Грид-технологии и облачные вычисления»

на тему

Реализация топологии «бабочка» для преобразования Фурье

Вариант № 2.6

Выполнили студенты Краснов А.П., Становова К.В.

Группа 6227М

Руководители работы Проценко В.И., Серафимович П.Г.

САМАРА 2015

содержание

[Задание 3](#_Toc408345852)

[Описание предметной области 3](#_Toc408345853)

[Описание инструментария 4](#_Toc408345854)

[Описание архитектуры 4](#_Toc408345855)

[Диаграмма потоков данных 5](#_Toc408345856)

[Приложение А 7](#_Toc408345857)

[Приложение Б 8](#_Toc408345858)

[Список использованных источников 10](#_Toc408345859)

# Задание

Реализовать топологию «бабочка» для Фурье преобразования произвольного N. Комментарии: perl mixed mode. Запустите программу с . Показать, что алгоритм работает правильно: обратное преобразование Фурье над полученными выходными значениями дает исходные значения. Для изображения «бабочки» при  составить изображение топологии программы (какие действия производятся в каждом операторе).

# Описание предметной области

Быстрое преобразование Фурье (БПФ) – алгоритм быстрого вычисления дискретного преобразования Фурье (ДПФ). Спектр последовательности  длины , , имеет следующий вид:

 (1)

где  – фазовый множитель, который обладает следующими свойствами:

1. , ;
2. ;
3. ;
4. ;

, , ,



Последняя операция называется «бабочка», она является основой приведенного метода.

Почти так же вычисляется обратное БПФ, с той лишь разницей, что нужно применить операцию комплексного сопряжения сначала к входной последовательности, а затем к выходной и все ее элементы разделить на . Схема работы метода для случая  представлена на рисунке 1.

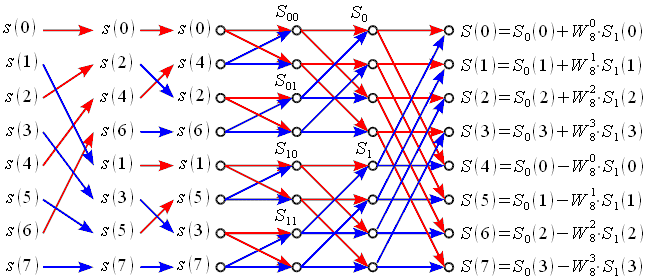


Рисунок 1 – Граф алгоритма БПФ с прореживанием по времени при 

# Описание инструментария

IBM InfoSphere Streams – вычислительная платформа, предназначенная для потоковой обработки данных. Данная программа установлена на виртуальной машине с операционной системой RedHat. В ней используется язык программирования IBM Streams Processing Language (IBM SPL). Среда разработки – InfoSphere Streams Studio. На рисунке 2 представлен пример работы со средой разработки.

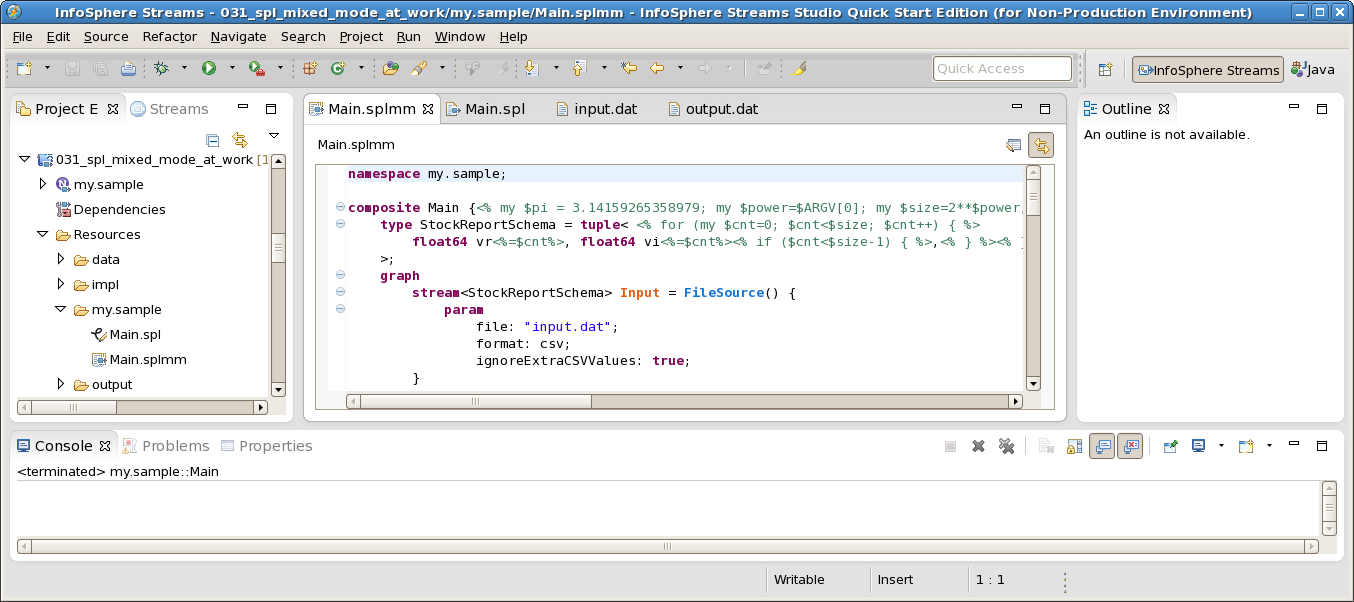


Рисунок 2 – Пример работы со средой разработки IBM InfoSphere Streams Studio

# Описание архитектуры

В данной программе используется Perl mixed mode – вставки в SPL кода на языке Perl. Этот режим применяется для генерации однотипного SPL-кода в зависимости от значений аргументов запуска программы.

Оператор Input = FileSource() считывает файл и передает в выходной поток числа, записанные в этом файле. Числа записаны в виде вещественной и мнимой частей, разделенных запятыми, как и сами числа. Используемые параметры: file – имя входного файла, format – его формат, ignoreExtraCSVValues – опция, позволяющая игнорировать пробелы, отступы и т.п.

Операторы Permutation1 = Custom(Input) и Permutation<%=$i%> = Custom(Permutation<%=$i-1%>) необходимы для осуществления двоично-инверсной перестановки входных значений. Для обратного преобразования в операторе Permutation1 также происходит замена исходных значений на комплексно сопряженные им числа.

Операторы Butterfly1 = Custom(Permutation<%=$power-1%>) и Butterfly<%=$i%> = Custom(Butterfly<%=$i-1%>) применяются для реализации операций «бабочка», которые приводят к конечному результату.

Оператор Output = Custom(Butterfly<%=$power%>) необходим при выполнении обратного преобразования для деления всех значений, рассчитанных на предыдущих этапах, на число элементов в последовательности, а также для замены полученных значений на комплексно сопряженные им числа.

Оператор FileWriter1 = FileSink(Output) печатает рассчитанные данные в выходной файл. Используемый параметр: file – имя выходного файла.

В качестве входных аргументов программы используется два параметра препроцессора mixed-mode (см. рисунок 3). Один из них (power) равен степени, в которую нужно возвести 2, чтобы получить размер обрабатываемых последовательностей. Для примера выберем этот аргумент равным 3, что соответствует 8 элементам в каждой последовательности. Другой (inverse) задает направление преобразования. Если данный аргумент равен 1, то выполняется обратное БПФ, иначе – прямое.

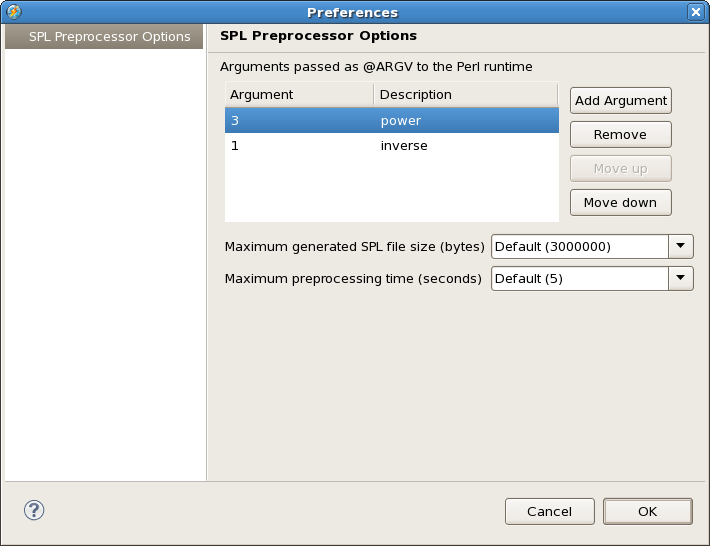


Рисунок 3 – Опции препроцессора mixed mode

# Диаграмма потоков данных

На рисунке 4 представлена диаграмма потоков данных (Data Flow Diagram), описывающая работу представленной программы для обратного преобразования последовательностей длины 8.

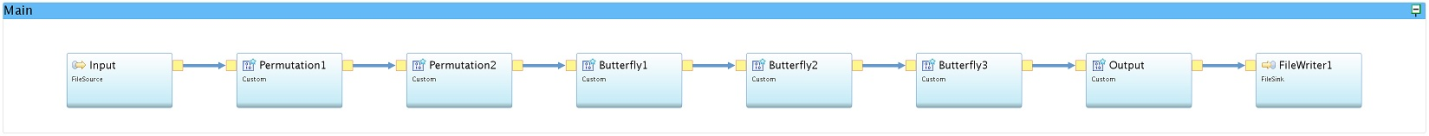


Рисунок 4 – Диаграмма потоков данных

На рисунке 5 изображена топология программы – каждому оператору сопоставлены действия, которые он производит при выполнении преобразования.

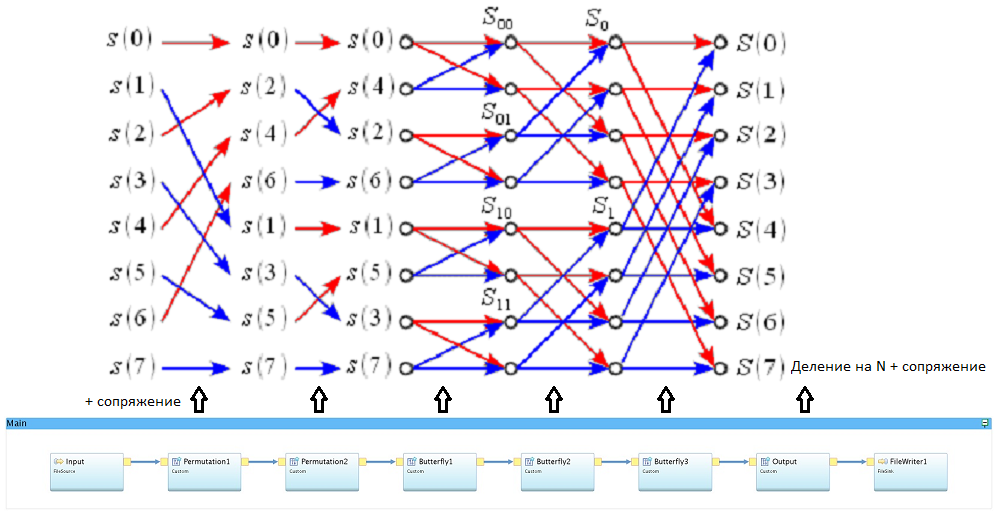


Рисунок 5 – Топология программы при  и обратном БПФ

В приложении А представлены результаты работы прямого БПФ над некоторыми входными последовательностями и обратного БПФ над результатами прямого преобразования. Нетрудно убедиться, что полученные последовательности совпадают в пределах допустимой погрешности (отклонения между числами по модулю не превышают ). Исходный код программы на языке SPL Mixed Mode приведен в приложении Б.

Приложение А

Входные данные (файл input.dat).

1, -1.5, 2, -1.3, 3, -1.1, 4, -0.9, 5, -0.7, 6, -0.5, 7, -0.3, 8, -0.1

-1, 0.1, -2, 0.3, -3, 0.5, -4, 0.7, -5, 0.9, -6, 1.1, -7, 1.3, -8, 1.5

-3, 5.4, 4, 2.0, 4.5, 2.1, -1.3, 2, 7, 0.1, -9.5, 4, 4.8, -5.4, 3.2, 2.9

8.1, 8, -5.6, 9, -2, 3.3, -4.6, 5.8, 1.1, 1.7, 1.2, -3.4, 5.6, -2.3, 4.3, 3.1

Выходные данные прямого БПФ (файл output.dat).

36, -6.4, -5.93137084989849, 8.85685424949238, -4.80000000000001, 3.2, -4.33137084989849, 0.856854249492373, -4, -0.800000000000001, -3.66862915010152, -2.45685424949238, -3.19999999999999, -4.8, -2.0686291501015, -10.4568542494924

-36, 6.4, 2.06862915010154, -10.4568542494924, 3.20000000000001, -4.8, 3.66862915010153, -2.45685424949238, 4, -0.800000000000001, 4.33137084989847, 0.856854249492386, 4.79999999999999, 3.2, 5.93137084989845, 8.85685424949238

9.7, 13.1, 8.17731239591687, -1.5417784899841, -4.20000000000001, 16.2, -32.2785317267988, -0.586143571373774, 16.9, -8.7, -13.1773123959169, 12.7417784899841, -6.39999999999999, 1.4, -2.72146827320116, 10.5861435713738

8.1, 25.2, 24.7622366364086, 31.8605122421383, 2.29999999999999, 12.8, 10.5923881554251, 2.94264068711929, 17.5, -3.8, 0.437763363591392, -4.06051224213832, 8.90000000000001, 4.6, -7.79238815542508, -5.54264068711931

Выходные данные для обратного БПФ, произведенного над значениями, которые были получены в ходе прямого БПФ.

1, -1.5, 2, -1.3, 3, -1.1, 4, -0.899999999999997, 5, -0.699999999999997, 6, -0.499999999999997, 7, -0.300000000000001, 8, -0.100000000000001

-1, 0.0999999999999983, -2, 0.3, -3, 0.500000000000001, -4, 0.700000000000001, -5, 0.900000000000002, -6, 1.1, -7, 1.3, -8, 1.5

-3, 5.4, 4, 2.00000000000001, 4.50000000000001, 2.09999999999999, -1.3, 2, 7, 0.099999999999997, -9.5, 3.99999999999999, 4.79999999999999, -5.39999999999999, 3.2, 2.9

8.1, 7.99999999999999, -5.6, 9, -2, 3.3, -4.6, 5.8, 1.1, 1.7, 1.19999999999999, -3.4, 5.6, -2.3, 4.3, 3.1

Приложение Б

Текст программы на языке SPL Mixed Mode (файл Main.splmm).

namespace my.sample;

composite Main {<% my $pi = 3.14159265358979; my $power=$ARGV[0]; my $inverse=$ARGV[1]; my $size=2\*\*$power; %>

type StockReportSchema = tuple< <% for (my $cnt=0; $cnt<$size; $cnt++) { %>

float64 vr<%=$cnt%>, float64 vi<%=$cnt%><% if ($cnt<$size-1) { %>,<% } %><% } %>

>;

graph

stream<StockReportSchema> Input = FileSource() {

param

file: "input.dat";

format: csv;

ignoreExtraCSVValues: true;

}

stream<StockReportSchema> Permutation1 = Custom(Input) {

logic

onTuple Input: {

StockReportSchema srs = {<% for (my $k=0; $k<2\*\*($power-1); $k++) { %><% my $b=$k+2\*\*($power-1); my $c=2\*$k; my $d=$c+1; %>

vr<%=$k%> = Input.vr<%=$c%>,

vi<%=$k%> = <% if ($inverse==1) { %>-<% } %>Input.vi<%=$c%>,

vr<%=$b%> = Input.vr<%=$d%>,

vi<%=$b%> = <% if ($inverse==1) { %>-<% } %>Input.vi<%=$d%><% if ($d<$size-1) { %>,<% } %><% } %>

};

submit(srs, Permutation1);

}

}

<% for (my $i=2; $i<$power; $i++) { %>

stream<StockReportSchema> Permutation<%=$i%> = Custom(Permutation<%=$i-1%>) {

logic

onTuple Permutation<%=$i-1%>: {

StockReportSchema srs = {<% for (my $j=0; $j<2\*\*($i-1); $j++) { %><% for (my $k=0; $k<2\*\*($power-$i); $k++) { %><% my $a=2\*\*($power-$i+1)\*$j+$k; my $b=$a+2\*\*($power-$i); my $c=2\*(2\*\*($power-$i)\*$j+$k); my $d=$c+1; %>

vr<%=$a%> = Permutation<%=$i-1%>.vr<%=$c%>,

vi<%=$a%> = Permutation<%=$i-1%>.vi<%=$c%>,

vr<%=$b%> = Permutation<%=$i-1%>.vr<%=$d%>,

vi<%=$b%> = Permutation<%=$i-1%>.vi<%=$d%><% if ($d<$size-1) { %>,<% } %><% } %><% } %>

};

submit(srs, Permutation<%=$i%>);

}

}

<% } %>

stream<StockReportSchema> Butterfly1 = Custom(Permutation<%=$power-1%>) {

logic

onTuple Permutation<%=$power-1%>: {

StockReportSchema srs = {<% for (my $j=0; $j<2\*\*($power-1); $j++) { %><% my $a=2\*$j; my $b=$a+1; %>

vr<%=$a%> = Permutation<%=$power-1%>.vr<%=$a%> + Permutation<%=$power-1%>.vr<%=$b%>,

vi<%=$a%> = Permutation<%=$power-1%>.vi<%=$a%> + Permutation<%=$power-1%>.vi<%=$b%>,

vr<%=$b%> = Permutation<%=$power-1%>.vr<%=$a%> - Permutation<%=$power-1%>.vr<%=$b%>,

vi<%=$b%> = Permutation<%=$power-1%>.vi<%=$a%> - Permutation<%=$power-1%>.vi<%=$b%><% if ($b<$size-1) { %>,<% } %><% } %>

};

submit(srs, Butterfly1);

}

}

<% for (my $i=2; $i<=$power; $i++) { %>

stream<StockReportSchema> Butterfly<%=$i%> = Custom(Butterfly<%=$i-1%>) {

logic

onTuple Butterfly<%=$i-1%>: {

StockReportSchema srs = {<% for (my $j=0; $j<2\*\*($power-$i); $j++) { %><% for (my $k=0; $k<2\*\*($i-1); $k++) { %><% my $p=2\*$pi\*$k\*2\*\*($power-$i)/$size; my $a=2\*\*$i\*$j+$k; my $b=$a+2\*\*($i-1); %>

vr<%=$a%> = Butterfly<%=$i-1%>.vr<%=$a%> + ((float64) <%=cos($p)%>) \* Butterfly<%=$i-1%>.vr<%=$b%> + ((float64) <%=sin($p)%>) \* Butterfly<%=$i-1%>.vi<%=$b%>,

vi<%=$a%> = Butterfly<%=$i-1%>.vi<%=$a%> - ((float64) <%=sin($p)%>) \* Butterfly<%=$i-1%>.vr<%=$b%> + ((float64) <%=cos($p)%>) \* Butterfly<%=$i-1%>.vi<%=$b%>,

vr<%=$b%> = Butterfly<%=$i-1%>.vr<%=$a%> - ((float64) <%=cos($p)%>) \* Butterfly<%=$i-1%>.vr<%=$b%> - ((float64) <%=sin($p)%>) \* Butterfly<%=$i-1%>.vi<%=$b%>,

vi<%=$b%> = Butterfly<%=$i-1%>.vi<%=$a%> + ((float64) <%=sin($p)%>) \* Butterfly<%=$i-1%>.vr<%=$b%> - ((float64) <%=cos($p)%>) \* Butterfly<%=$i-1%>.vi<%=$b%><% if ($b<$size-1) { %>,<% } %><% } %><% } %>

};

submit(srs, Butterfly<%=$i%>);

}

}

<% } %><% my $div=1; if ($inverse==1) { $div=$size; } %>

stream<StockReportSchema> Output = Custom(Butterfly<%=$power%>) {

logic

onTuple Butterfly<%=$power%>: {

StockReportSchema srs = {<% for (my $cnt=0; $cnt<$size; $cnt++) { %>

vr<%=$cnt%> = Butterfly<%=$power%>.vr<%=$cnt%> / ((float64) <%=$div%>),

vi<%=$cnt%> = <% if ($inverse==1) { %>-<% } %>Butterfly<%=$power%>.vi<%=$cnt%> / ((float64) <%=$div%>)<% if ($cnt<$size-1) { %>,<% } %><% } %>

};

submit(srs, Output);

}

}

() as FileWriter1 = FileSink(Output) {

param

file: "output.dat";

}

}

Список использованных источников

1. IBM Knowledge Center [Интернет-ресурс]. URL: <http://www-01.ibm.com/support/knowledgecenter/>, дата доступа: 01.12.2014.
2. БПФ по основанию 2 с прореживанием по времени [Интернет-ресурс]. URL: <http://www.dsplib.ru/content/thintime/thintime.html>, дата доступа: 01.12.2014.
3. Методы компьютерной обработки изображений / Под ред. В.А. Сойфера. М.: ФИЗМАТЛИТ, 2003. 784 с.