Статическая балансировка нагрузки в параллельной реализации алгоритма частотного анализа текстовой информации

Ба Хла Тхан, С.А. Лупин, Ай Мин Тайк, Хейн Тун

***Аннотация* — Обсуждается эффективность статической балансировки нагрузки в параллельной реализации алгоритма частотного анализа текстовой информации. Алгоритм реализован в виде многопоточного приложения. Сравниваются два метода распределения вычислений между потоками – с учетом частотных характеристик букв и без нее. Экспериментальные результаты показали, что учет частоты повторяемости букв позволяет ускорить работу приложения при анализе, как монограмм, так и биграмм.**

***Ключевые слова* — частотный анализ текста; многопоточные приложения; балансировка нагрузки вычислительных узлов.**

I. Введение

Текстовый анализ широко используется для исследования основных характеристик различных текстов. Социальные и гуманитарные науки изучают характеристики текста, чтобы раскрыть различия и сходства между разными культурами народов и субкультурами, а также выявить их исторические корни. Важными текстовыми характеристиками являются: повторяемость букв и их пар, совместимость букв друг с другом, чередование гласных и согласных, и некоторые другие особенности. Эти характеристики достаточно стабильны в большинстве текстов. Частотность букв сильно зависит от длины анализируемого текста и его характера. Стабильными для языка являются характеристики, которые получены при анализе большого количества различных текстов.

Частотный анализ текстов находит широкое применение не только в лингвистике. Развитие информационных технологий приводит к появлению новых задач, не имеющих аналогии в других областях. При работе различных клиентских и поисковых служб у провайдеров интернет сервисов формируются log-файлы, содержащие различную информацию о параметрах клиентских запросов. Как правило, это текстовые файлы и для их анализа может быть использован частотный метод.

Статья получена 10 июня 2016. Статья подготовлена в рамках гранта РФФИ № 16-07-01055\165 "Адаптация ресурсоемких алгоритмов к распределенной вычислительной среде".

Ба Хла Тхан, аспирант, Национальный исследовательский университет «МИЭТ», (e-mail: bahlathan51@gmail.com);
Ай Мин Тайк, аспирант, Национальный исследовательский университет «МИЭТ», (e-mail: ayeminthike52@gmail.com);
Хейн Тун - аспирант, Национальный исследовательский университет «МИЭТ», (e-mail: lighter2k@gmail.com);
С. А. Лупин, профессор, Национальный исследовательский университет «МИЭТ», (e-mail: lupin@miee.ru).

Важной особенностью такого подхода является то, что он позволяет проводить и ретроспективный анализ данных, используя архивные файлы разных форматов. Это дает возможность исследовать изменчивость параметров клиентских запросов и проводить превентивный реинжиниринг сервисов.

II. Текстовые данные

Текстовые данные являются богатейшим источником социальной информации, важным ресурсом для социальных аналитиков. Разнообразие текстовых источников позволяет интерпретировать и описывать нормы общественного поведения, общественные структуры и ценности. Анализ текста играет важную роль в социальных науках. Сегодня появился широкий спектр методов обработки текстов, кодирования, создания схем категоризации. Эти методы используются не только для описания и кодирования отношений между различными текстовыми элементами, но и для исследования больших объемов текстовых данных.

Классификация текстовых данных, которые можно использовать для социологического анализа, весьма широка. В основе деления может лежать происхождение теста: ответы на вопросы анкет, газетные и журнальные статьи и комментарии, различные виды отчетов, служебные записки, репортажи, рекламные объявления, стенограммы выступлений и бесед, интервью, письма. Такая классификация является весьма условной и открытой, поскольку установить точные границы классов невозможно.

В зависимости от целей анализа, текстовые данные могут относиться к одной семантической области, например, только политические новости, или могут быть связаны с несколькими семантическими областями. Кроме того, они могут состоять из одного конкретного типа текста, например, только личные письма или включать в себя тексты, относящиеся ко всему авторскому наследию.

Важным фактором для скомпилированных текстов является их принадлежность одному или нескольким авторам. Кроме того, тексты, принадлежащие одному автору, могут включать цитаты из чужих произведений.

Знание типа анализируемого текста необходимо при выборе методов его обработки и консолидации, которые предшествуют фазе анализа. Выборка источников и формирование из них связанного текста определяют необходимую полноту анализа. Стратегии подбора анализируемых источников непосредственно связанны не только с целями анализа, но и с методами адекватного отображения его результатов [1].

III. Частотный анализ текстовой информации

Частотный анализ текстовой информации является важной частью исследований в таких областях как вычислительная лингвистика, обработка естественного языка, классификация и категоризация текстов, анализ авторства [2].

Количество текстовой информации в электронной форме постоянно увеличивается, поэтому методы категоризации текста должны обеспечивать нахождение релевантной информации в различных источниках при выполнении таких операций, как поиск ответов, классификация новостей или групп новостей, сортировка сообщений электронной почты.

Задача классификации текста заключается в его ассоциации с одной или несколькими категориями из заданного набора. Методы категоризации отличаются формами представления документов и способами определения категорий конкретного документа. Как правило, для каждого типа текста может быть подобран метод категоризации, наиболее полно учитывающий особенности данных, свойственных этому типу. Мера, которая используется для количественной оценки эффективности каждого метода, также влияет на результаты сравнения [3]. Рассмотрим несколько классических методов классификации, опирающихся на частотный анализ.

Существуют несколько основных подходов к проблеме классификации языков [4]. Простейший подход заключается в формировании лексического словаря для каждого возможного языка, с целью последующего сравнения таких словарей с анализируемым текстом. При этом текст будет отнесен к тому языку, элементы лексического словаря которого, наиболее часто встречаются в нем.

Несмотря на кажущуюся простоту, формирование репрезентативного лексического словаря не является тривиальной задачей, особенно для некоторых из малоиспользуемых языков. Кроме того, если язык имеет множество склонений, то есть в нем используются различные формы слов, то приходится либо формировать очень большие словари, чтобы обеспечить включение в них всех форм слов, или применять некоторые средства для специфической морфологической обработки текста для уменьшения числа различных форм слов. Если текст является результатом процесса OCR (Optical Character Recognition), то в нем могут быть ошибки распознавания из-за плохого качества изображения, и они будут нарушать релевантность процесса классификации.

Другой подход к классификации языка заключается в использование для анализа N-грамм (сочетаний из N букв). Основная идея заключается в подсчете в тексте частоты появлении N-грамм. Такие частоты являются убедительным доказательством идентификации текста на принадлежность его к определенному языку. Техника сравнения частотных профилей N-грамм может использоваться для языковой классификации документов. Она не требует предварительного построения лексических словарей или морфологической обработки текста. Более того, релевантная классификация может быть получена и для относительно небольших текстов – от 10K до 20K байт. Рассчитанные для них профили частот N-грамм, гораздо более информативны, чем результаты лексического анализа [5,6].

Эффективность метода классификации с использованием N-грамм была протестирована на примере идентификации языка сообщений электронной почты, циркулирующих в новостных группах Usenet. Эти сообщения были написаны на различных языках, но с помощью нескольких преобразований для обработки диакритических маркировок все были представлены в стандартной кодировке ASCII. В зависимости от длины сообщения правильно был определен язык от 92.9 до 99.8% текстов.

Основные этапы процедуры классификации:

- Сформировать обучающую текстовую выборку для каждого языка, на котором может быть написано сообщение. Как правило, эти выборки имеют объем от 20K до 120K байт, не требующие предварительного форматирования текста.
- Вычислить частотный профиль N-грамм в обучающей текстовой выборке.
- Вычислить частотный профиль N-грамм сообщения. Полученный профиль имеет объем порядка 4К.
- Вычислить общую меру расстояния между профилем исследуемого сообщения и профилем для каждого языка. Для классификации языковой принадлежности сообщения используется категория с наименьшим расстоянием.

Большинство современных исследований в области анализа текстовой информации, например [7,8], связаны с адаптацией модели MapReduce к вычислению N-грамм, в данной работе основной акцент направлен на рассмотрение возможности повышения эффективности параллельной реализации процедуры построения частотных профилей за счет статической балансировки нагрузки.

IV. Параллельная обработка данных

Большинство современных компьютеров могут обеспечить параллельные вычисления благодаря архитектуре микропроцессоров. Параллельные вычисления хорошо подходят для решения сложных задач, включая и анализ данных. Их основные преимущества связанны с сокращением времени вычислений. Это позволяет использовать вычислительно сложные алгоритмы и увеличивать размерность решаемых задач.

С другой стороны, параллельные вычисления значительно увеличивают сложность приложений. Кроме того, все известные алгоритмы должны быть существенно изменены для того, чтобы реализовывать их на параллельном компьютере. Если алгоритмы не будут адаптированы к параллельной среде,

эффективность их параллельных реализаций будет низкая.

Существуют несколько методов программирования, которые можно использовать для создания параллельного приложения:

- Разделяемая память без потоков;
- Разделяемая память с потоками (OpenMP);
- Распределенная память с мультипрограммированием (MPI);
- Гибридные приложения (MPI + OpenMP).

Для распределения работы в параллельных приложениях используются два традиционных подхода и их комбинация:

- Распределение частей кода между обработчиками;
- Распределение данных между обработчиками;
- Сочетание обоих этих методов.

Разработанное приложение использует технологию OpenMP для построения многопоточного приложения и разделение данных между потоками.

V. Программная реализация алгоритма и результаты тестов

Параллельная программа для исследования эффективности статической балансировки нагрузки узлов при решении задачи частотного анализа реализована как многопоточное приложение в среде Intel Parallel Studio (IPS) с использованием библиотеки OpenMP [9].

В качестве тестовых задач были использованы тексты на английском языке, содержащие от $5*10^7$ до 10^8 букв. Приложение ориентировано на построение частотных профилей двух видов – для монограмм и для биграмм.

В табл. 1 показано статистическое распределение букв в текстах на английском языке. При распределении нагрузки между ядрами использовано два метода.

Таблица 1 *Частота повторения букв*

Символ	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*
Частота	8.2	1.5	2.8	4.3	12.7	2.2	2.0	6.1	7.0
Символ	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*
Частота	0.2	0.8	4.0	2.4	6.8	7.5	1.9	0.1	6.0
Символ	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*	
Частота	6.3	9.1	2.8	1.0	2.4	0.2	2.0	0.1	

Первый не учитывает статистическую повторяемость букв, при этом они распределяются между потоками в алфавитном порядке.

Время решения задачи для такого подхода обозначено в табл. 4 и 5 как t_1. Анализ работы приложения, выполненный средствами IPS (рис. 1), показал, что имеется неравномерность загрузки ядер, приводящая к низкой эффективности приложения.

Второй метод распределения нагрузки основан на статической балансировке нагрузки узлов [10]. При таком подходе буквы алфавита распределяются между потоками так, чтобы суммарные частоты выделенных потокам литер были близки. Вариант сбалансированного распределения нагрузки для различного числа потоков представлен в табл. 2.

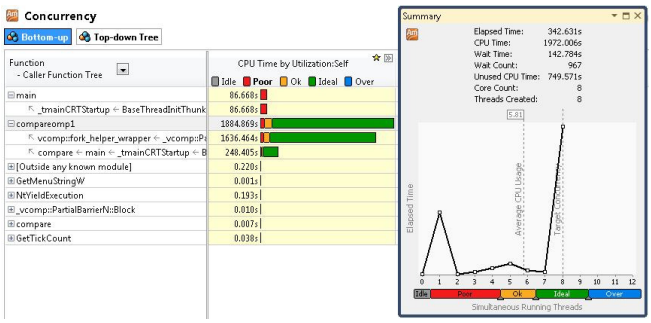

Рис. 1. Эффективность работы многопоточного приложения анализа монограмм

Таблица 2 *Статическая балансировка нагрузки*

Число потоков	*Распределение букв между потоками*								
2	t a i s h l c f w p q x j	e z o n r d u m y g b k v							
3	y o s d b y p k z	a i r l m w c x j	e n h u f g v q						
4	o r u y q j h	z t n d p g k	a s l f w v	e i b c m x					
5	t q h k w y	d i r b v	n c m z o	a s g x u	e l p f j				
6	a l u b j	t d y v z	n h g p	o r f k	i s c x	e m q w			
7	a l w x	t f j c	i d v b	e g z q	u n p m	r y s	o h k		
8	a w x g	t y v z	o f c	e q k	l i b	r d p	h j s	n u m	

Принцип распределения нагрузки в многопоточных приложениях, использующих функцию *omp parallel for*, позволяет оценить максимально возможное ускорение вычислений. Такая оценка базируется на соотнесении числа итераций внешнего цикла (26 для английского алфавита) с числом инициируемых потоков. Например, для трех потоков эта оценка составит 3*(26/27), а для шести потоков – 6*(26/30), поскольку максимально возможное ускорение может быть получено при условии постоянной загрузки всех имеющихся ядер. Результаты оценки представлены в табл. 3.

Таблица 3 *Оценка максимального ускорения*

Число потоков	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Ускорение	1	2	2,89	3,71	4,33	5,2	6,5	6,5

Добиться абсолютного равенства нагрузки в задаче частотного анализа не удается, поскольку относительно небольшое число букв в алфавите нельзя равномерно распределить между потоками.

Таблица 4 *Частотный анализ монограмм*

Число потоков	*Размер задачи*							
	$5*10^7$				10^8			
	t_1	S_1	t_2	S_2	t_1	S_1	t_2	S_2
1	3,77	1	3,77	1	7,49	1	7,49	1
2	1,89	1,99	1,88	2	3,79	1,97	3,77	1,98
3	1,35	2,79	1,29	2,92	2,71	2,76	2,59	2,89
4	1,05	3,59	1,00	3,77	2,11	3,55	2,01	3,72
5	0,91	4,14	0,85	4,43	1,83	4,09	1,70	4,41
6	0,78	4,83	0,72	5,24	1,56	4,8	1,44	5,20

7	0,62	6,08	0,59	6,39	1,23	6,08	1,17	6,4
8	0,60	6,28	0,58	6,50	1,21	6,18	1,15	6,5

Однако и приведенные в табл. 2 наборы букв позволяют повысить эффективность работы приложения. Время решения задачи для такого подхода обозначено в табл. 4 и 5 как t_2. Символом S обозначено ускорение вычислений по отношению к однопоточному варианту программы.

При анализе монограмм статическая балансировка нагрузки демонстрирует положительный эффект во всем диапазоне проводимых исследований (рис. 2).

Таблица 5 *Частотный анализ биграмм*

Число потоков	*Размер задачи*							
	*5*10^7*				*10^8*			
	t_1	S_1	t_2	S_2	t_1	S_1	t_2	S_2
1	69,78	1	69,78	1	138,54	1	138,54	1
2	35,24	1,98	34,78	2	70,93	1,95	69,42	2
3	25,95	2,69	23,91	2,91	52,24	2,65	47,82	2,89
4	19,91	3,5	18,38	3,79	39,92	3,47	36,70	3,77
5	17,56	3,98	15,16	4,6	35,36	3,92	30,31	4,57
6	15,19	4,59	13,06	5,33	30,37	4,56	26,08	5,31
7	11,84	5,89	10,91	6,38	23,72	5,84	21,59	6,42
8	11,74	5,94	10,62	6,56	23,63	5,86	21,16	6,55

В ряде случаев достигнутое ускорение превышает максимальную оценку. Это объясняется тем, что потоки загружены неравномерно и тот, который раньше завершит выполнение итерации, сразу начинает выполнение следующей.

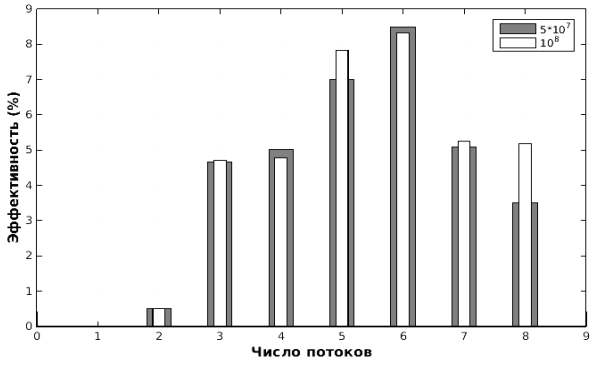

Рис. 2. Эффективность статической балансировки при анализе монограмм

С формальной точки зрения, при анализе биграмм статическая балансировка нагрузки должна основываться именно на их повторяемости, но это слишком громоздкая статистика. Поэтому при распределении нагрузки мы использовали тот же метод, что и в случае монограмм, т.е. табл. 2. Представленные в табл. 5 результаты подтверждают эффективность подобного подхода. Использование статической балансировки для анализа биграмм дает даже больший эффект, чем для монограмм (рис. 3).

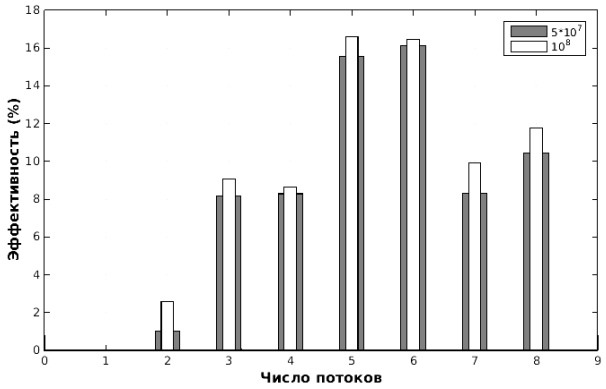

Рис. 3. Эффективность статической балансировки при анализе биграмм

Представленные выше результаты были получены при запуске приложения на двухпроцессорном вычислительном узле с процессорами Intel Xeon E5335 (2.0 GHz, quad-core). Общее число доступных приложению ядер равно восьми. Отметим, что эти процессоры не поддерживают технологию гипертрединга, поэтому в экспериментах число потоков было ограничено восемью.

Сегодня значительный интерес вызывают вопросы применимости ускорителей Intel Xeon Phi для реализации различных алгоритмов. В таблице 6 представлены результаты запуска приложения на ускорителе Intel® Xeon Phi™ Coprocessor 7120, имеющим 61 ядро. В качестве тестовых задач были использованы тексты на английском языке, содержащие 10^8 и 10^9 букв.

Нагрузка равномерно распределялась между ядрами ускорителя без учета частоты повторения литер. Полученные результаты достаточно очевидны - максимальное ускорение достигается в том случае, когда число букв в алфавите кратно числу потоков.

Таблица 6 *Результаты вычислений на Intel Xeon Phi*

Число потоков	10^8		10^9	
	t	*S*	t	*S*
1	5,62	1,0	56,26	1,0
2	2,81	2,0	28,16	2,0
3	1,95	2,9	19,48	2,9
4	1,52	3,7	15,17	3,7
5	1,30	4,3	12,99	4,3
6	1,09	5,2	10,81	5,2
7	0,87	6,4	8,66	6,5
8	0,87	6,4	8,66	6,5
9 - 12	0,66	8,5	6,50	8,7
13 - 25	0,45	12,4	4,31	13,0
26	0,24	23,4	2,17	25,9
52	0,16	35,57	1,13	49,79

В рассматриваемом алгоритме достаточно сложно реализовать статическую балансировку нагрузки для

большого числа потоков, которые можно запускать на ускорителе. Тем не менее, мы можем воспользоваться данными (табл. 1) для распределения нагрузки. Полученные результаты представлены в таблице 7.

Таблица 7 *Балансировка нагрузки на Intel Xeon Phi*

Число потоков	10^8		10^9	
	t	*S*	t	*S*
Ядра распределяем пропорционально частоте букв				
52	0,26	21,62	2,2	25,57
60	0,26	21,62	2,2	25,57
Ядра распределяем поровну и добавляем еще по одному ядру наиболее часто встречающимся литерам				
60	0,15	37,4	1,12	50,23

В первых двух случаях статическая балансировка оказывает отрицательное влияние на производительность приложения, поскольку время работы определяется самой медленной ветвью программы. Кратное увеличения числа ядер в данном случае не оправдано. В последнем случае эффект от балансировки более ощутим, однако результат лишь немного превосходит равномерное распределение (табл. 6).

Таблица 8 *Анализ биграмм на Intel Xeon Phi*

Число потоков	*Размер задачи*							
	$5*10^7$				10^8			
	t_1	S_1	t_2	S_2	t_1	S_1	t_2	S_2
1	76,51	1	76,51	1	152,79	1	152,79	1
*5*8*	2,33	32,84	2,31	33,12	4,59	33,29	4,57	33,43
*8*5*	2,51	30,48	2,41	31,75	4,85	31,5	4,75	32,17
*6*7*	2,22	34,46	2,19	34,94	4,35	35,12	4,34	35,21
*7*6*	2,12	36,09	2,07	36,96	4,1	37,27	4,12	37,09
*6*8*	1,98	38,64	1,96	39,04	3,92	38,98	3,81	40,10
*8*6*	2,19	34,93	2,02	37,88	4,09	37,36	4,08	37,45
*7*8*	1,66	46,09	1,62	47,23	3,2	47,75	3,13	48,81
*8*7*	1,82	42,04	1,77	43,23	3,63	42,09	3,57	42,8

При анализе биграмм на ускорителях Intel Xeon Phi мы можем использовать различные подходы к балансировке нагрузки. В табл. 8 приведены результаты, которые позволяют оценить эффективность сочетания двух методов распределения нагрузки. В столбце «число потоков» первая цифра соответствует одному из сбалансированных вариантов распределения алфавита между потоками (табл. 2), а вторая цифра показывает, на сколько частей разделяется сам анализируемый текст. Число генерируемых в приложении потоков равно произведению этих двух цифр.

Время решения задачи при использовании статической балансировки (t_2), сравнивается с вариантом распределения неупорядоченного алфавита (t_1). Во всех случаях статическая балансировка нагрузки позволяет получить большее значение ускорения (*S*), что говорит о ее эффективности.

Отметим также, что для ускорителей Intel Xeon Phi статическая балансировка нагрузки, представляет значительно больший интерес, чем динамическая. Это определяется особенностями их архитектуры. Загрузка исполняемого ядрами ускорителя кода через шину PCI, что характерно для динамической балансировки, требует значительных временных затрат, а при статической балансировке загрузка кода производится только однократно – при запуске приложения

VI. Заключение

Балансировка нагрузки узлов является одним из методов повышения эффективности параллельных приложений. Его использование в приложении для частотного анализа текста позволяет сократить время решения задачи почти на 10% (табл. 4, 5, 8), что можно считать значимым результатом для приложений, оптимизированных в среде Intel Parallel Studio.

Использованный метод разделения нагрузки применим только для небольшого числа потоков, поскольку предельное число потоков не может превосходить числа букв в алфавите. При этом нагрузка потоков будет несбалансированной. Решением этой проблемы может стать более сложный метод разделения нагрузки, предусматривающий и разделение не только алфавита, но и текста между потоками. Например, частоту литеры «е» будут подсчитывать 5 потоков, разделяющих между собой текст, а для литеры «а» будет выделено только 3 потока. Такой подход позволит повысить масштабируемость приложения и обеспечить баланс загрузки узлов. Конечно, он должен использоваться для больших объемов текста. Дальнейшие исследования будут направлены на экспериментальную проверку этого предположения.

Литература

[1] C. Graesser, Danielle S. Mcnamara, Max M. Louwerse, Zhiqiang Cai. Coh-Metrix: Analysis of text on cohesion and language // Behavior Research Methods, Instruments, & Computers, Vol.5, 2004.

[2] C Namrata Mahender. Text classification and classifiers: a survey // International Journal of Artificial Intelligence & Applications (IJAIA), Vol.3, No.2, 2012.

[3] Ana Cardoso-Cachopo, Arlindo Limede Oliveira. An Empirical Comparison of Text Categorization Methods // In Proceedings of 10-th International Symposium, SPIRE 2003, Manaus, Brazil, October 8-10, 2003.

[4] D. Graus, T. Kenter, M. Bron, E. Meij, M. de Rijke. Context-based entity linking. // In Proc. of Fifth Text Analysis Conference (TAC 2012), Gaithersburg, MA, 2012, 13 p.

[5] William B. Cavnar, John M. Trenkle. N-Gram-Based Text Categorization // In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, 1994.

[6] Florian Beil, Martin Ester, Xiaowei Xu. Frequent Term-Based Text Clustering // In Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD 2002)

[7] Klaus Berberich, Srikanta Bedathur. Computing n-Gram Statistics in MapReduce // In Proceedings of the 16th International Conference on Extending Database Technology (EDBT '13), pp. 101-112

[8] Samuel Huston, Alistair Moffat, W. Bruce Croft. Efficient Indexing of Repeated n-Grams // In Proceedings of the Fourth ACM International Conference on Web Search and Data Mining (WSDM 2011

[9] OpenMP. http://www.openmp.org

[10] Бо Тянь, М.А. Посыпкин, И.Х. Сигал. "Балансировка нагрузки на основе оценок алгоритмической сложности подзадач" // Информационные технологии и вычислительные системы, №1, с. 10–18, 2015.

Static Load Balancing in Parallel Algorithm of the Frequency Analysis of Textual Information

Ba Hla Than, S. Lupin, Aye Min Thike, Hein Tun

Abstract— **This paper discusses the efficiency of static load balancing in a parallel implementation of the algorithm of frequency analysis of textual information. The algorithm is implemented as a multi-threaded application. We have compared two methods of job distribution between flows – with accounting the frequency characteristics of letters and without it. In this paper, we showed experimental results that the accounting of letters repetition frequency allows speeding up the work of application in the analysis both monograms and bigrams.**

Keywords — frequency *analysis of text; multi-threaded applications; load balancing of computing nodes.*