УДК 004.054

И. В. ГРУЗДО, С. В. РОССОХА, И. В. ШОСТАК

Национальный аэрокосмический университет им. Н. Е. Жуковского «ХАИ», Украина

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КОМПЬЮТЕРИЗАЦИИ ПРОЦЕССА УСТАНОВЛЕНИЯ АВТОРСТВА ТЕКСТА НА ОСНОВЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

В работе проведено решение задачи преобразования последовательных программ в параллельные. Обсуждены типичные элементы распараллеливания, используемые в современных системах. Проведена методика поиска заимствований и определения авторства текстов в виде последовательного выполнения операций. Сформулирована задача распараллеливания для процесса анализа заимствований в конкретном типе письменных учебных работ. Дано описание параллельного алгоритма для процесса выполнения анализа заимствований и определения авторства текстов, позволяющих за приемлемое время получать распределения параллельных ветвей анализа текстов письменных учебных работ по процессорным ядрам.

Ключевые слова: *параллельные вычисления, плагиат, анализ, распараллеливание, преобразование, распределенная память, степени параллелизма.*

Введение

На современном этапе развития информационных технологий, как во всем мире, так и в Украине особую актуальность приобретают задачи, связанные с машинным поиском информации [1, 2]. К этому классу задач относится и компьютеризация процессов анализа различного рода письменных документов с целью выявления в них заимствований. Одной из типичных задач указанного класса является автоматический поиск заимствований во всякого рода письменных учебных работах.

Процесс выявления заимствований в письменных учебных работах (ПУР) состоит из следующих этапов [3]: определение специфики учебных работ, диагностика причин заимствования, классификация видов заимствований ПУР, оценивание текстов ПУР с позиций наличия в них плагиата, создание технологии выявления заимствований в письменных учебных работах, организация машинного анализа ПУР на предмет наличия в них плагиата с целью повышения качества учебного процесса в целом.

Реализация перечисленных выше этапов связана с решением еще одной важной прикладной задачи, а именно со структурированием анализируемой ПУР, установлением семантических связей между отдельными структурными частями с учетом их значимости.

В свою очередь, при решении данной задачи необходимо сравнивать письменную учебную работу со значительным количеством проверенных работ, находящихся в хранилище, что приводит к увеличению времени на выполнение процесса анализа.

Указанные выше обстоятельства определяют актуальность задачи разработки эффективного информационного инструментария для контроля качества выполнения академических работ на предмет выявления плагиата с использованием параллельных вычислений, что даст возможность снизить затрачиваемое время на поиск заимствований.

Цель статьи состоит в изложении основных этапов преобразования последовательных программ в параллельные, а также описание параллельного алгоритма и принципов функционирования программного средства, реализующего указанную технологию.

Решение задачи преобразования последовательных программ в параллельные

Задача преобразования ПС «Plagiarizm» в параллельную программу может быть представлена в виде следующего набора подзадач:

1. Распараллеливание линейных участков. Линейным участком программы назовем часть программы, операторы которой выполняются в естественном порядке или в порядке, определенном командами безусловных переходов. Линейный участок ограничен начальным и конечным операторами. Внутри линейного участка распараллеливание может производиться по операторам, а внутри операторов - распараллеливание арифметических выражений. Распараллеливание по операторам позволяет, при наличии вычислительных ресурсов, обеспечить реализацию одновременно нескольких операторов, что ускоряет проведение вычислений.

2. Распараллеливание циклов. Границы циклов определяются по формальным признакам описания

© И. В. Груздо, С. В. Россоха, И. В. Шостак

циклов в соответствующих языках программирования. Процесс преобразования последовательного цикла для параллельного выполнения заключается в том, что из переменных формируются векторы, над которыми выполняются векторные операции.

3. Распараллеливание по процессам. В вычислительных системах с множественным потоком команд одновременно может выполняться несколько участков программ. В зависимости от структуры вычислительных систем взаимосвязь потоков команд между собой может быть различна. Каждый поток команд может быть совершенно независимой задачей. Потоки команд могут образовывать подзадачи одной задачи. В последнем случае между подзадачами сохраняется связь по данным, но каждая подзадача – это, как правило, достаточно большая программа, слабо связанная с другими подзадачами по управлению.

Разбиение программы на асинхронные параллельно выполняемые процессы может оказаться существенным фактором ускорения вычислительного процесса.

4. Разбиение программы на слабо связанные участки. Такое разбиение может оказаться полезным в тех случаях, когда анализ всей программы потребует слишком много времени из-за огромного числа связей и комбинаций, которые необходимо проверить. Предварительная разбивка программ на слабо связанные участки, т.е. декомпозиция программ, может сократить это время, хотя, конечно, повлияет на полноту распараллеливания. Метод разбиения программ на слабо связанные участки может оказаться эффективным при разбиении программ по узлам (вычислительным машинам) локальной сети. Он может позволить сократить число межмашинных обменов.

В соответствии с наданной функциональной декомпозицией, преобразование ПС «Plagiarizm» в параллельную программу предполагает присвоение разным потокм данных различных функций, но в силу того, что множество задач выполняются последовательно, целесообразно провести распараллеливание на уровне анализа ПУР.

Для того, чтобы преобразовать ПС «Plagiarizm» в параллельную программу рассмотрим основные этапы разработки параллельного ПО. Процесс распараллеливания можно разделить на две части: анализ исходной программы; синтез параллельной программы.

Анализ исходной программы включает в себя: анализ задачи с целью выделить подзадачи, которые могут выполняться одновременно; выявление основных типичных элементов и представление связей между ними или пошаговых действий.

Следует отметить, что по своей природе ПС «Plagiarizm» относится к событийно-ориентированным приложениям, и, кроме того, имеет сложно организованную структуру, когда в составе программы наряду с параллельными имеются и циклические участки.

Методика поиска заимствований и определения авторства текстов в виде последовательного выполнения операций

Формально задача выявления текстологических заимствований в различных типах учебных работ относится к классу задач принятия решений. Ограничениями являются вид поиска (точный или нечеткий). Результатом работы модели применение в ходе поиска последовательности методов, рациональных для поиска заимствований в конкретном типе письменных учебных работ.

Методика поиска заимствований в ПУР с учетом типа и значимости структурных частей ПУР предусматривает реализацию следующих этапов.

1 Преобразование анализируемой ПУР в вид, удобный для построения полнотекстового индекса, вычисляемого автоматически для каждой рубрики (выделение из документов содержательной информационной основы).

2 Выполнение автоматической классификации анализируемой ПУР по матрице согласования приоритетов уровней:

а) вычисление признакового пространства $\Pi = \Pi_1 \times \Pi_2 \times \ldots \times \Pi_N$, i-го признака Π_i рассматриваемой ПУР;

б) нахождение признакового описания рассматриваемой ПУР pc_i по заданному в матрице согласования приоритетов уровней;

в) определение отношения $\Phi : D \times C \to \{0,1\}$, для каждой пары (pc_i, cp_j), относится ли данная ПУР pc_i, имеющая признаковое описание $f(pc_i)$, к категории cp_j. Для этого находим из матрицы максимальную близкую к функции Φ функцию $\Phi_{ПУР}$. Каждое значение $f(pc_i)$ нормируется от нуля до единицы. Поэтому и расстояние нормировано к единице.

3 Нахождение множества ПУР в банке данных работ, соответствующих типу анализируемой ПУР.

4 Выбор поискового инструментария из следующей номенклатуры методов поиска α_1 – Рабина; α_2 – Бойера-Мура-Хорспула; α_3 – Маасса-Новака; α_4 – Левенштейна; α_5 – Ханта-Шиманского; α_6 – Машека-Патерсона.

5 Вычисление заимствования по выбранному методу с учетом значения условий перехода:

$$\alpha_1 = \begin{cases} 1- \text{ если } X = (K = ПУР_{лаб}) \vee (R = 1) \vee (10 < V < 16) \vee (S = 1) \| \\ \qquad (K = ПУР_{реф}) \vee (R = 0) \vee (10 < V < 16) \vee (S = 1) \| \\ \qquad (K = ПУР_{дз}) \vee (R = 0) \vee (10 < V < 16) \vee (S = 1); \\ 0 - \text{если нет}; \end{cases}$$

$$\alpha_2 = \begin{cases} 1- \text{ если } X = (K = ПУР_{дз}) \vee (R = 1) \vee (15 < V < 60) \vee (S = 1) \| \\ \qquad (K = ПУР_{реф}) \vee (R = 1) \vee (10 < V < 60) \vee (S = 1) \| \\ \qquad (K = ПУР_{вр}) \vee (R = 0) \vee (50 < V < 60) \vee (S = 1); \\ 0 - \text{если нет}; \end{cases}$$

$$\alpha_3 = \begin{cases} 1- \text{ если } X = (K = ПУР_{вр}) \vee (R = 1) \vee (60 < V < 120) \vee (S = 1) \| \\ \qquad (K = ПУР_{дп}) \vee (R = 0) \vee (60 < V < 120) \vee (S = 1) \| \\ \qquad (K = ПУР_{др}) \vee (R = 0) \vee (60 < V < 120) \vee (S = 1); \\ 0 - \text{если нет}; \end{cases}$$

$$\alpha_4 = \begin{cases} 1- \text{ если } X = (K = ПУР_{лаб}) \vee (R = 1) \vee (10 < V < 16) \vee (S = 0) \| \\ \qquad (K = ПУР_{кп}) \vee (R = 0) \vee (10 < V < 16) \vee (S = 0) \| \\ \qquad (K = ПУР_{дз}) \vee (R = 0) \vee (10 < V < 16) \vee (S = 0) \| \\ \qquad (K = ПУР_{др}) \vee (R = 0) \vee (60 < V < 120) \vee (S = 0); \\ 0 - \text{если нет}; \end{cases}$$

$$\alpha_5 = \begin{cases} 1- \text{ если } X = (K = ПУР_{дз}) \vee (R = 1) \vee (15 < V < 50) \vee (S = 0) \| \\ \qquad (K = ПУР_{кп}) \vee (R = 1) \vee (10 < V < 40) \vee (S = 0) \| \\ \qquad (K = ПУР_{кр}) \vee (R = 1) \vee (10 < V < 40) \vee (S = 0) \| \\ \qquad (K = ПУР_{вр}) \vee (R = 0) \vee (50 < V < 60) \vee (S = 0); \\ 0 - \text{если нет}; \end{cases}$$

$$\alpha_6 = \begin{cases} 1- \text{ если } X = (K = ПУР_{вр}) \vee (R = 1) \vee (60 < V < 120) \vee (S = 0) \| \\ \qquad (K = ПУР_{дп}) \vee (R = 1) \vee (60 < V < 120) \vee (S = 0) \| \\ \qquad (K = ПУР_{др}) \vee (R = 1) \vee (60 < V < 120) \vee (S = 0); \\ 0 - \text{если нет}. \end{cases}$$

6 Формирование списка фрагментов заимствований из других текстов ПУР, при этом индекс каждого элемента списка будет указывать на структурную часть анализируемой ПУР.

Качественным отличием описанной методики анализа текстовых документов является учет особенностей, присущих письменным учебным работам, что, в конечном итоге, приводит к повышению эффективности организации учебного процесса в заведениях различного уровня аккредитации.

Описание процесса выполнения анализа заимствований и определения авторства текстов в форме параллельных вычислений

Сформулируем задачу распараллеливания для процесса анализа заимствований в конкретном типе письменных учебных работ.

Имеется упорядоченная схема ПУР в виде множества вершин V, соответствующих типу ПУР, которые пронумерованы от 1 до n. Т.е задана схема и взаимно однозначное отображение $V_{нум} \to \{1,...,n\}$.

Требуется найти такое отображение $V_{скорект} : \{1,...,n\} \to \{1,...,m\}$, чтобы выполнялось следующее условие.

Управляющая схема, получающаяся из исходной нумерации, должна быть эквивалентной. Т.е. требуется, чтобы вычисления на ней оставались бы ровно в тех случаях в которых определены на предыдущем шаге для анализа ПУР, и были соотнесены с теми же результатами к значениям приписанным в начале анализа вершинам соответствующего типа и значения. При этом никак не должны учитываться свойства вызываемых процедур поискового инструментария.

Введем оператор R раскрытия состояния программы: если $X=(t, i, S)$ — произвольное, отличное от финального состояние программы (т.е. $S \subset O_n \setminus \{o(i)\}$, то $R=(t, i, S)$ - множество состояний, порождаемых состоянием X, т.е. непосредственно следующих за этим состоянием. Раскрыв начальное состояние A_0 и последовательно раскрывая все далее получаемые состояния, мы в конечном итоге дойдем до совокупности финальных состояний программы во время процесса анализа вида $(t, i, O_n \setminus \{o(i)\})$. Определим также оператор R^*, который, в отличие от оператора R, раскрывает не состояния, а их расширения - записи. Записью будем называть кортеж объектов подлежащих анализу (t,i,S,z,z^*,c,h,v), где $(t,i,S)=X$ - состояние программы, z - получаемый номер этого состояния, z^* - номер непосредственно предшествующего состояния, c - стоимость, определяемая построенной совокупностью записей траектории от начального состояния до состояния X, h и v - соответственно нижняя и верхняя оценки стоимостей всех возможных полных траекторий из A_0 в финальные состояния программы, начальная часть которых совпадает с траекторией из A_0 в X.

Нижняя оценка h есть сумма $c+\sum^H(t,i,O_n \setminus S)$, где через $\sum^H(t,i,O_n \setminus S)$ обозначена сумма $\sum_{o(j)\in S} \phi_{min}(t,i,j)$, в которой $\phi_{min}(t,i,j)=\phi\left(i, t_{min}^*(t,i,j)\right)$ - величина штрафа по объекту анализа o(j) при условии, что его обслуживание начинается вслед за объектом анализа o(i), т.е. в момент времени $t_{min}(t,i,j)=\max(t+l(i,j), t(j))$; при этом $t_{min}^*(t,i,j)$ - момент завершения анализа o(j), т.е. $t_{min}(t,i,j)+\tau(j, t_{min}(t,i,j))$. Очевидно, что в силу монотонности функции штрафа, величина $\sum^H(t,i,O_n \setminus S)$ является неуменьшаемой нижней оценкой оптимального штрафа по объектам множества $O_n \setminus S$.

Верхняя оценка v есть сумма $c+\sum^V(t,i,O_n \setminus S)$, где $\sum^V(t,i,O_n \setminus S)$ - верхняя оценка оптимального суммарного штрафа по объектам множества $O_n \setminus S$ при их анализе вслед за o(i). Значение последней предлагается вычислять согласно следующей пошаговой процедуры, относящейся к классу жадных алгоритмов [5].

1. Производится инициализация первоначальных значений переменных алгоритма: $\bar{S}=O_n \setminus S, \bar{i}=i, \bar{t}=t, \sum^V(t,i,O_n \setminus S)=0$.

2. В качестве очередного объекта, принимаемого к анализу, выбирается тот объект $o(k) \in \bar{S}$, который обеспечивает минимум приращения нижней оценки суммарного штрафа по объектам множества $\bar{S} \setminus \{o(k)\}$:

$$\min_{k\in S}\left\{\sum^H(t^*(\bar{t},\bar{i},k),k,\bar{S} \setminus \{o(k)\}) - \sum^H(t^*(t,\bar{i},\bar{S} \setminus \{o(k)\})\right\}$$

3. С учетом номера выбранного объекта производится обновление значений переменных алгоритма:

$$\bar{S}=\bar{S} \setminus \{o\{k\}\},$$

$$\bar{t}=t_{min}{}^*(t,\bar{i},k), \sum^V(t,i,O_n \setminus S)=\sum^V(t,i,O_n \setminus S)+$$

$$+\phi_{min}(\bar{t},\bar{i},k), \bar{i}=k$$

4. Если $\bar{S}=\varnothing$, то переход на шаг 2, в противном случае - останов.

Будем говорить, что запись $Z=(t,i,S,z,z^*,c,h,v)$ доминируется записью $Z_1=(t_1,i,S_1,z_1,z_1{}^*,c_1,h_1,v_1)$, если $t_1 \le t$, $S \subseteq S_1$, $c_1 \le c$, причем, по меньшей мере, одно из записанных соотношений выполняется как строгое (строгое включение или строгое неравенство). Очевидно, что в случае, когда запись Z_1 доминирует запись Z, траектория, заканчивающаяся в итоге расширением Z, предпочтительнее траектории, заканчивающейся расширением Z. Очевидно также, что если существует траектория, заканчивающаяся расширением $Z_2=(t_2,i,S_2,z_2,z_2{}^*,c_2,h_2,v_2)$, и такая, что $v_{2\le h}$, то в этом случае состояние X (соответствующее расширению Z) является бесперспективным направлением счета и может быть исключено из дальнейшего рассмотрения.

Итак, предполагаем, что параллельная система (ПС) состоит из m вычислительных узлов $cl_1,...,cl_m$ и одного узла-координатора cl_0, соединенных локальной сетью.

1. В начальный момент времени все вычислительные узлы находятся в режиме ожидания, т.е.:

– массив записей, раскрываемых узлом cl_i - пуст $(M_i=\varnothing)$;

– массив всех сгенерированных параллельной системой записей, о которых на данный момент времени известно узлу cl_i, а также массив номеров записей, помеченных как удаленные, пусты: $G_i=\varnothing$, $D_i=\varnothing$ $(i=\overline{1,m})$.

– множество номеров свободных узлов на координаторе содержит номера всех узлов ПС $(J = \{1,2,...,m\})$, а массивы всех сгенерированных системой нефинальных и финальных записей, а также записей, помеченных как удаленные - пусты: $G = \varnothing,\ F = \varnothing,\ D = \varnothing$.

2. Координатор отправляет на cl_1 начальную запись $Z_0 = (0,0,\varnothing,0,0,0,h_0,\nu_0)$ отражая изменение состояния системы в массивах J и G. $J = \{2,3,...,m\}, G = \{Z_0\}$.

3. На каждом узле cl_i:

– если узел находится в режиме ожидания $(W_i = 0)$ и получает от координатора пакет записей T, подлежащих раскрытию, то этот пакет переносится в массив $M_1 (M_1 = T)$ и узел переводится в рабочий режим $(W_i = 1)$;

– если узел находится в рабочем режиме $(W_i = 1)$:

– если $M_i = \varnothing$, то узел переводится в режим ожидания $(W_i = 0)$ и об этом сообщается координатору $(J = J \cup \{i\})$;

– если $M_i \neq \varnothing$, начинает действовать основная часть алгоритма.

3.1. Узлом запрашиваются от координатора записи из массива G, начиная с $\left(|G_i|+1\right)$-й и добавляются в конец G_i; аналогичным образом запрашиваются номера удаленных записей начиная с $\left(|D_i|+1\right)$-го элемента массива D и добавляются в конец D_i.

3.2. Из M_i изымаются все записи, к ним применяется оператор R^* и результат сохраняется в G_i^*.

3.3. Из числа записей массива G_i помечаются как удаленные те записи, которые либо доминируются записями из G_i^*, либо их нижняя оценка превышает верхнюю для какой-либо записи множества G_i^*. Совокупность записей, помеченных к удалению, добавляется в конец D_i.

3.4. Если в G_i^* есть финальные записи, то они перемещаются в F_i.

3.5. Устанавливается соединение с координатором, в рамках которого передается содержимое массивов D_i, G_i^* и F_i и сообщается о незанятости узла $cl_i (J = J \cup \{i\})$.

3.6. Узел переводится в режим ожидания, т.е. $G_i^* = \varnothing,\ F_i = \varnothing,\ F_i = 0$.

4. При установлении соединения по инициативе узла cl_i координатор принимает результаты работы узла (массивы D_i, G_i^* и F_i) и добавляет их в конец соответствующих совокупных массивов записей, сгенерированных всей ПС

$$(D = D \cup D_i,\ G = G \cup G_i^*,\ F = F \cup F_i).$$

После этого в массиве F оставляются только записи с минимальным значением суммарного штрафа, а остальные записи удаляются. По завершении соединения принятая координатором совокупность нефинальных записей G_i^* распределяется между узлами множества J, и соответствующие узлы изымаются из J. Заметим, что соединения узлов с координатором осуществляются в монопольном режиме с целью синхронизации процесса обновлений массивов D, G и F.

5. Процесс завершается, когда $J = \{1,2,...,m\}$ и $G_i^* = \varnothing$ $(i = \overline{1,m})$, т.е. все узлы завершили вычисления и нераскрытые нефинальные записи отсутствуют. В этом случае массив F содержит все финальные записи с минимальным значением штрафа c. Определив траекторию системы, ведущую из начального состояния в состояние, определяемое некоторой записью массива F, очевидным образом можно построить соответствующую оптимальную стратегию.

Приведенный параллельный алгоритм поиска заимствований и определения авторства текстов, а также разработанная группа алгоритмов распараллеливания, позволяют за приемлемое время получать распределения параллельных ветвей анализа текстов ПУР по процессорным ядрам.

Выводы

Рассмотрены существующие типы распараллеливания приложений и проведен их анализ.

Приведено описание параллельного алгоритма поиска заимствований и определения авторства текстов, а также разработана группа алгоритмов, позволяющих за приемлемое время получать распределения параллельных ветвей анализа текстов ПУР по процессорным ядрам. Первая группа алгоритмов относится к классу вероятностных алгоритмов локальной оптимизации, вторая основана на алгоритмах обхода графов и их разбиении.

Литература

1. Груздо, І. В. Анализа программного обеспечения для обнаружения плагиата в научных работах [Текст] / І. В. Груздо // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління : 1-я науч.-техн. конф. 13-14 грудня 2010. – К. : НАУ, 2010. – С. 264.

2. Груздо, И. В. Проблемы анализа естественно-языковых текстов для обнаружения плагиата в учебных работах [Текст] / И. В. Груздо // Радіоелектронні і комп'ютерні системи. – 2011. – №1 (49). – С. 130- 138.

3. Груздо, И. В. Информационная технология выявления заимствований в работах студентов технических вузов на основе нечеткого поиска [Текст] / И. В. Груздо, И. В. Шостак // Информационные системы и технологии ИСТ-2012 : Межнар. науч.-техн. конф. посвященная 75-летию В. В. Свиридова. – Х., 2012. – С. 31.

4. Ситкевич, Т. А. Параллельные вычислительные среды [Текст] : учеб.-метод. пособие / Т. А. Ситкевич, В. Н. Сюрин. – Гродно, 2001. - 115 с.

5. Минаев, Д. В. Идея параллельного алгоритма синтеза стратегий обслуживания группировки стационарных объектов [Текст] / Д. В. Минаев, Ю. С. Федосенко, А. Ю. Шлюгаев // Высокопроизводительные параллельные вычисления на кластерных системах : 7-я Межнар. конф.-сем. – Нижний Новгород : Изд-во Нижегородского госуниверситета, 2007. - С. 248-256.

Поступила в редакцию 5.06.2014, рассмотрена на редколлегии 17.06.2014

Рецензент: д-р техн. наук, проф., зав. каф. экономики и маркетинга В. М. Вартанян, Национальный аэрокосмический университет им. Н. Е. Жуковского «ХАИ», Харьков.

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КОМП'ЮТЕРИЗАЦІЇ ПРОЦЕСУ ВСТАНОВЛЕННЯ АВТОРСТВА ТЕКСТУ НА ОСНОВІ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ

І. В. Груздо, С. В. Россоха, І. В. Шостак

У роботі наведено рішення задачі перетворення послідовних програм в паралельні. Обговорено типові елементи розпаралелювання, що використовуються в сучасних системах. Приведено методику пошуку запозичень і визначення авторства текстів у вигляді послідовного виконання операцій. Сформульовано завдання розпаралелювання для процесу аналізу запозичень в конкретному типі письмових навчальних робіт. Дано опис паралельного алгоритму для процесу виконання аналізу запозичень і визначення авторства текстів, які дозволяють за прийнятний час отримувати розподіл паралельних гілок аналізу текстів письмових навчальних робіт по процесорним ядрам.

Ключові слова: паралельні обчислення, плагіат, аналіз, розпаралелювання, перетворення, розподілена пам'ять, ступені паралелізму.

DEVELOPMENT OF SOFTWARE FOR THE COMPUTERIZATION OF ATTRIBUTION TEXT BASED ON PARALLEL

I. V. Gryzdo, S. V. Rossokha, I. V. Shostak

In the work the solution of converting serial programs in parallel. Discussed the typical elements of parallelization used in modern systems. Conducted search technique borrowing and authorship of texts as a series of operations. The problem of parallelization for the analysis process of borrowing in a particular type of written educational works. A description of the parallel algorithm for analysis of the implementation process of borrowing and authorship of texts allow a reasonable time to obtain the distribution of parallel branches analysis of texts written academic works on the processor cores.

Keywords: parallel computing, plagiarism, analysis, parallelization, transformation, distributed memory parallelism.

Груздо Ирина Владимировна – канд. техн. наук, ассистент кафедры инженерии программного обеспечения, Национальный аэрокосмический университет им. Н. Е. Жуковского «Харьковский авиационный институт», Харьков, Украина, e-mail: tigralwovna@rambler.ru.

Россоха Сергей Владимирович - канд. техн. наук, доцент кафедры инженерии программного обеспечения, Национальный аэрокосмический университет им. Н. Е. Жуковского «Харьковский авиационный институт», Харьков, Украина, e-mail: sergey.rossokha@gmail.com.

Шостак Игорь Владимирович – д-р техн. наук, проф., проф. кафедры инженерии программного обеспечения, Национальный аэрокосмический университет им. Н. Е. Жуковского «Харьковский авиационный институт», Харьков, Украина, e-mail: iv_shostak@rambler.ru.