УДК 004.633, 004.032.24, 004.032.34, 004.042

ОБРАБОТКА ОШИБОК В ИНТЕГРИРОВАННЫХ РАСЧЕТАХ

А.М. НАЗАРЕНКО[1,2], А.А. ПРОХОРОВ[1,2]
[1] ООО «ДАТАДВАНС», г. Москва, Россия
[2] Институт проблем передачи информации РАН им. А.А. Харкевича, г. Москва, Россия

Работа выполнена при финансовой поддержке РФФИ
в рамках научного проекта № 15-29-07043

Проводимые в ходе решения многодисциплинарных инженерных и научных задач вычислительные эксперименты требуют совместного применения нескольких расчетных средств. При этом в рамках таких действий, как выполнение заданного плана эксперимента или поиска оптимальных решений, производится многократный прогон одной последовательности вычислений с различными параметрами и входными данными. Для автоматизации подобных экспериментов необходимо реализовать схему расчета, управляющую запуском расчетных средств и передачей данных между ними. Как правило, такие схемы разрабатываются на базе специализированного программного обеспечения – интеграционной среды или интеграционной платформы. Результатом разработки является интегрированный расчет (платформозависимая реализация схемы расчета), который с точки зрения автоматизации представляет собой композицию слабо связанных (в смысле интенсивности взаимодействия) типовых подзадач. В такой композиции можно выделить ряд шаблонов расчетов (типов взаимодействия подзадач), которые в свою очередь могут рассматриваться как подзадачи уже более высокого порядка.

В настоящей работе рассматриваются вопросы управления исполнением и управления данными в среде исполнения интегрированных расчетов при обнаружении ошибки какого-либо из интегрируемых расчетных средств. Под ошибкой понимается любое внештатное поведение расчетного средства, приводящее к невозможности сформировать ожидаемые от него результаты расчета и таким образом нарушающее схему передачи данных в интегрированном расчете. В качестве основного требования к механизму обработки ошибок принимается требование не допустить некорректного завершения расчета в целом при отсутствии каких-либо промежуточных данных расчета. Формулируются правила обработки ошибок на уровне шаблонов расчетов и на уровне составного расчета, являющегося композицией шаблонов как подзадач. Отмечены случаи, в которых поведение шаблона расчета при ошибке может быть различным в зависимости от целей пользователя, и возможные варианты выбора поведения, которые могут быть указаны пользователем.

Ключевые слова: автоматизация расчетов, интеграция расчетов, обработка ошибок, поток работ, шаблоны потоков работ.

ВВЕДЕНИЕ

Вычислительная задача, требующая совместного применения нескольких расчетных средств (РС), может быть представлена как композиция типовых подзадач, или шаблонов расчетов. В настоящей работе рассматриваются вопросы обработки ошибок отдельных РС в таких задачах (интегрированных расчетах) на различных уровнях, а также сохранения расчетных данных в случае ошибки. Шаблоны расчетов здесь описываются только в той мере, которая необходима для обсуждения основных вопросов; подробное их описание приведено в [1].

Под ошибкой будем понимать любое поведение интегрируемых РС, приводящее к невозможности сформировать ожидаемые результаты расчета. Причины ошибок могут быть различны, например:

1) некорректное завершение РС вследствие программного или аппаратного сбоя;

2) отсутствие результатов при корректном завершении РС – например, в случае, когда входные данные не попадают в область определения расчетной модели (заранее не известную) и ее выходы не могут быть вычислены;

3) завершение РС пользователем – например, если какой-либо решатель в расчете расходится, пользователь может остановить его работу для освобождения вычислительных ресурсов.

Интегрированный расчет, как правило, выполняется в специализированной интеграционной среде – в настоящее время существует уже целый ряд подобных систем, в частности [2–6]. Одной из задач среды исполнения и является предоставить механизм обработки ошибок [7–9]. Ключевое требование к данному механизму – ошибка РС не должна приводить к безусловной остановке расчета. Далее в настоящей работе указывается, как должны обрабатываться ошибки в шаблонах расчетов, чтобы ошибка отдельного РС в составе шаблона не приводила к некорректному завершению расчета. Также одна из особенностей расчетов, проводимых в ходе какого-либо вычислительного эксперимента, состоит в том, что полезным результатом работы расчета являются не только данные, полученные от последнего запущенного РС, а все данные и файлы, полученные в процессе работы всех входящих в состав расчета РС. Например, в инженерном расчете некоторого изделия характеристики, вычисленные для каждого из узлов конечно-элементной сетки, являются полезным результатом работы расчета, наряду с полученными интегральными характеристиками модели изделия. Поэтому в дополнение к правилам обработки ошибок далее отмечается и порядок сохранения результатов в случае возникновения ошибки. При этом, как и в [1], для простоты изложения мы считаем, что каждое РС выдает результаты своей работы в виде файлов, которые находятся в рабочей директории РС. Правила сохранения результатов, таким образом, можно считать дополнением к правилам обработки файлов, описанным в [10, 11].

ОБРАБОТКА ОШИБОК В ПОСЛЕДОВАТЕЛЬНОМ РАСЧЕТЕ

В последовательном расчете происходит поочередный запуск РС, и выходные данные каждого РС являются необходимыми для запуска следующего РС.

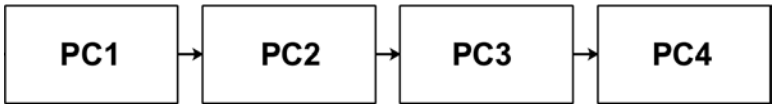

Рис. 1. Пример схемы последовательного расчета
Fig. 1. Example of a sequential calculation scheme

В общем случае некоторые РС в последовательном расчете могут быть модулями одной системы, которым требуется общая рабочая директория. Остальные РС могут выполняться изолированно. Например, для расчета на рис. 1 структура рабочих директорий может иметь следующий вид:

- Расчет/ – директория с результатами расчета
  - PC1/ – рабочая директория PC1
  - Система_PC/ – общая рабочая директория PC2 и PC3
  - PC4/ – рабочая директория PC4

Последовательный расчет останавливается при ошибке любого РС, поскольку результаты работы данного РС напрямую или опосредованно требуются для запуска всех последующих. При этом результаты уже отработавших РС сохраняются в соответствующих директориях. Если последовательный расчет является частью составного расчета, то его остановка по ошибке РС в свою очередь обрабатывается как ошибка уже на уровне составного расчета.

ОБРАБОТКА ОШИБОК В ПАРАЛЛЕЛЬНОМ РАСЧЕТЕ

В параллельном расчете два или более РС получают данные из одного источника и работают независимо, а для получения итогового результата необходимы выходные данные всех таких РС.

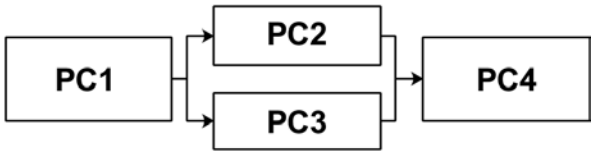

Рис. 2. Пример схемы параллельного расчета
Fig. 2. Example of a parallel calculation scheme

Поскольку параллельно работающие РС независимы и запускаются одновременно, все они должны исполняться в изолированных рабочих директориях. Например, структура рабочих директорий для расчета на рис. 2:

- Расчет/ – директория с результатами расчета
  - PC1/
  - …
  - PC4/ – рабочие директории РС

Параллельный расчет также останавливается при ошибке любого РС, поскольку для получения конечного результата расчета необходим полный набор результатов РС. Особенность его в том, что при параллельном исполнении независимых РС ошибка одного из них никак не влияет на другие. Управление параллельными РС в таком случае может быть разным.

1. Поскольку итоговый результат расчета уже не может быть получен (отсутствуют данные РС, завершившегося с ошибкой), производится остановка всех РС для экономии ресурсов и остановка расчета.

2. Остановка расчета откладывается, пока не завершат работу остальные параллельные (независимые) РС. В примере на рис. 2 такая ситуация возможна при ошибке РС2. Такое поведение является компромиссом между экономией ресурсов и сохранением рабочих данных и промежуточных результатов. Отметим, что в общем случае возможна ситуация, когда некоторые параллельные РС еще не запустились к моменту возникновения ошибки. Здесь также возникают варианты: либо такие РС уже не запускаются, несмотря на наличие входных данных для них, либо остановка расчета откладывается, пока не будут исполнены все параллельные РС.

Таким образом, ошибка в параллельном расчете всегда приводит к ошибке самого расчета (в смысле определения ошибки, данного во введении), однако остановка параллельного расчета по ошибке может быть отложена. В любом случае, результаты работы корректно завершившихся РС сохраняются в их рабочих директориях. В случае, когда параллельный расчет является составным (например, одна из независимых ветвей представляет собой последовательный расчет), появляются и другие особенности обработки ошибок – они будут обсуждаться ниже при рассмотрении составного расчета.

ОБРАБОТКА ОШИБОК В АЛЬТЕРНАТИВНОМ РАСЧЕТЕ

В альтернативном расчете, как и в параллельном, присутствуют несколько РС, получающих данные из одного источника, однако при каждом прогоне расчета работает только одно из них. Этот выбор зависит от входных данных расчета.

Рис. 3. Пример схемы альтернативного расчета
Fig. 3. Example of an alternative calculation scheme

Альтернативно запускаемые РС используют общую рабочую директорию, прочие запускаются изолированно. Для примера на рис. 3:

- Расчет/ – директория с результатами расчета
  - РС1/ – рабочая директория РС1
  - РС2/ – общая рабочая директория РС2а, РС2б
  - РС1/ – рабочая директория РС1

Поскольку в альтернативном расчете нет одновременно исполняемых РС, правила обработки ошибок в нем аналогичны последовательному расчету: ошибка любого РС приводит к немедленной остановке расчета; текущие результаты сохраняются в рабочих директориях РС; остановка обрабатывается как ошибка на уровне составного расчета, если он существует.

ОБРАБОТКА ОШИБОК В ПАКЕТНОМ РАСЧЕТЕ

В пакетном расчете одно РС используется для обработки нескольких наборов входных данных. Запуски экземпляров РС с разными наборами входных данных могут происходить и последовательно, и параллельно.

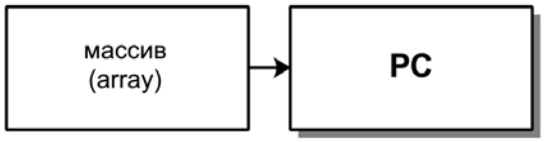

Рис. 4. Схема пакетного расчета
Fig. 4. Batch calculation scheme

Каждый набор данных обрабатывается независимо, экземпляры РС запускаются в изолированных рабочих директориях. Структура хранения файлов должна определять, к какому набору входных данных относится каждая из рабочих директорий, например:

- Расчет/ – директория расчета
  - 1/ – рабочая директория экземпляра РС, обрабатывающего 1-й набор
  - 2/ – 2-й набор и т. д.
  - …

Экземпляры РС работают независимо, при этом в пакетном расчете результаты их работы уже не используются. Следовательно, при ошибке любого из них (ошибке обработки очередного набора) расчет может продолжиться, просто перейдя к следующему набору. Конечный результат расчета в таком случае будет включать неполный набор данных. Однако если пакетный расчет является элементом составного расчета, возможна и ситуация, когда зависящим от него расчетам требуется только полный набор результатов. Следовательно, поведение пакетного расчета при ошибке РС может быть различным.

1. Расчет не останавливается, происходит переход к следующему набору входных данных. Результат расчета включает неполные данные.

2. Расчет останавливается, рабочие данные РС сохраняются, однако результат расчета не формируется.

Первый вариант поведения соответствует корректному завершению пакетного расчета, т. е. ошибка не доходит до уровня составного расчета. При втором варианте остановка пакетного расчета обрабатывается как ошибка на уровне составного расчета.

ОБРАБОТКА ОШИБОК В ИТЕРАТИВНОМ РАСЧЕТЕ

Итеративный расчет также использует одно РС для обработки нескольких наборов входных данных, однако их количество не определено до начала исполнения, и в общем случае входные данные для последующих расчетов (итераций) зависят от результатов предыдущих. Завершение итеративного расчета зависит от критерия, рассматривающего результат очередной итерации или некую совокупность результатов проведенных итераций.

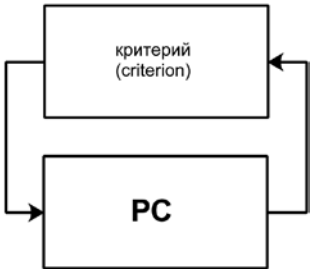

Рис. 5. Схема итеративного расчета
Fig. 5. Iterative calculation scheme

Структура хранения файлов, как и в пакетном расчете, должна сохранять соответствие директории с результатами определенному набору входных данных, т. е. определенной итерации. Итеративность позволяет в том числе просто нумеровать директории в хронологическом порядке, например:

- Расчет/ – директория расчета
  - 1/ –директория 1-й итерации
  - 2/ – директория 2-й итерации и т. д.
  - …

Правила обработки ошибок РС в итеративном расчете зависят от того, допускает ли критерий завершения отсутствие результатов отдельной итерации и могут ли входные данные для следующей итерации быть сформированы в таком случае. Варианты поведения аналогичны пакетному расчету: либо ошибка РС подавляется, формируется новый набор входных данных и начинается следующая итерация, либо происходит остановка итеративного расчета. При первом варианте поведения итеративный расчет всегда завершается корректно, при втором – остановка расчета может быть обработана как ошибка на уровне составного расчета.

ОБРАБОТКА ОШИБОК В СОСТАВНОМ РАСЧЕТЕ

Под составным понимается расчет, который является композицией вышеперечисленных шаблонов и может в свою очередь входить в любой из этих шаблонов как единое РС. В структуре хранения данных такой расчет всегда изолирован от других РС и составных расчетов.

Выше мы указывали возможное поведение шаблонных расчетов при возникновении ошибок составляющих их РС. Отдельный шаблонный расчет также может быть представлен составным расчетом, т. е. как единое РС. В случаях, когда ошибка РС приводит к остановке такого расчета, это представление дает возможность обрабатывать остановку точно так же, как ошибку РС, т. е. обеспечивает единообразие правил обработки ошибок в произвольных расчетных схемах.

В обсуждении обработки ошибок в параллельном расчете отмечалось, что расчет может потребовать остановки уже запустившихся РС при обнаружении ошибки. Такая ситуация воз-

можна и в пакетном расчете, если происходит параллельный запуск нескольких экземпляров РС. Однако в сложной расчетной схеме место РС в шаблоне может занять составной расчет, в свою очередь состоящий из нескольких РС. Вследствие этого появляются дополнительные требования к механизму обработки ошибок:

- помимо обработки ошибок, составляющих его РС, составной расчет должен обеспечивать обработку сигнала остановки от расчета, в который он входит как РС.
- вполне вероятно, что в момент получения составным расчетом сигнала остановки исполняется одно или несколько входящих в него РС. Значит, сигнал должен также указывать, каким образом осуществляется остановка, т. е. иметь две версии: «мягкая» остановка (составной расчет ожидает завершения тех РС в его составе, которые уже запущены) и немедленная остановка (все РС завершаются, несмотря на возможную потерю данных).

Как видно, этот же механизм остановки применим и к отдельным РС. Отметим, что остановка РС или составного расчета по сигналу не приводит к возникновению ошибки (невозможности формирования результатов), т. к. это действие уже есть реакция на возникновение ошибки. Таким образом, мы получаем единый механизм обработки ошибок в расчетных схемах, основанный на правилах обработки ошибок в типовых шаблонах расчетов и возможности представления расчета как РС.

ЗАКЛЮЧЕНИЕ

Для типовых шаблонов расчетов возможно указать правила обработки ошибок входящих в них РС, применимые как в отдельном расчете того или иного типа, так и в сложных расчетных схемах, являющихся композицией данных шаблонов. При этом в ряде случаев возможны различные варианты поведения, выбор между которыми нельзя сделать на основании структуры расчетной схемы или типа шаблона. Указать предпочтительный вариант поведения в таких случаях может только составляющий расчетную схему пользователь, основываясь на требованиях к сохранению данных расчета. Следовательно, в расчетной схеме должен существовать элемент, который с точки зрения пользователя является обработчиком ошибок и позволяет настраивать правила обработки. В качестве такого элемента удобно использовать составной расчет, который может включать в себя любой из типовых шаблонов, их композицию или даже единичное РС. Это дает пользователю возможность указания различных правил обработки ошибок на различных участках в сложных расчетных схемах путем их декомпозиции и представления в виде нескольких составных расчетов.

Благодарности
Условия для выполнения работ по проекту предоставлены ООО «ДАТАДВАНС».

СПИСОК ЛИТЕРАТУРЫ

1. Прохоров А.А., Назаренко А.М., Давыдов А.В. Шаблоны инженерных и научных расчетов // CEUR Workshop Proceedings, 2016, vol. 1787, pp. 403–409.

2. Радченко Г. И. Грид-система CAEBeans: интеграция ресурсов инженерных пакетов в распределенные вычислительные среды // Вестник Нижегородского университета им. НИ Лобачевского. 2009. №. 6 (1). С. 192–202.

3. Deelman E., Gannon D., Shields M., Taylor I. Workflows and e-Science: An overview of workflow system features and capabilities. Future Generation Computer Systems, 2009, vol. 25, no. 5, pp. 528–540.

4. Автоматизация инженерных расчетов, анализ данных и оптимизация с помощью программного комплекса PSE/MACROS / Е.В. Бурнаев, Ф.В. Губарев, С.М. Морозов, А.А. Прохоров, Д.С. Хоминич // Межотраслевая информационная служба. 2012. № 4 (165). С. 41–50.

5. Knyazkov K.V., Kovalchuk S.V., Tchurov T.N., Maryin S.V., Boukhanovsky A.V. CLAVIRE: e-Science infrastructure for data-driven computing. Journal of Computational Science, 2012, vol. 3, no. 6, pp. 504–510.

6. Sukhoroslov O., Volkov S., Afanasiev A. A Web-Based Platform for Publication and Distributed Execution of Computing Applications. 14th International Symposium on Parallel and Distributed Computing (ISPDC). IEEE, 2015, pp. 175–184.

7. Luo Z. et al. Exception handling in workflow systems. Applied Intelligence, 2000, vol. 13, no. 2, pp. 125–147.

8. Hagen C., Alonso G. Exception handling in workflow management systems. IEEE Transactions on software engineering, 2000, vol. 26, no. 10, pp. 943–958.

9. Russell N., van der Aalst W., ter Hofstede A. Workflow exception patterns. International Conference on Advanced Information Systems Engineering. Springer Berlin Heidelberg, 2006, pp. 288–302.

10. Nazarenko A.M., Prokhorov A.A. Hierarchical Dataflow Model with Automated File Management for Engineering and Scientific Applications. Procedia Computer Science, 2015, vol. 66, pp. 496–505.

11. Назаренко А.М., Пересторонин Н.О., Прохоров А.А. Управление файлами в рамках модели потоков данных для распределенных вычислений // Научный Вестник МГТУ ГА. 2016. Том 19, № 05. С. 161–172.

СВЕДЕНИЯ ОБ АВТОРАХ

Назаренко Алексей Михайлович, старший программист, ООО «ДАТАДВАНС»; младший научный сотрудник, Институт проблем передачи информации им. А.А. Харкевича РАН, alexey.nazarenko@datadvance.net.

Прохоров Александр Александрович, начальник отдела разработки ПО, ООО «ДАТАДВАНС»; научный сотрудник, Институт проблем передачи информации им. А.А. Харкевича РАН, alexander.prokhorov@datadvance.net.

ERROR HANDLING IN INTEGRATION WORKFLOWS

Alexey M. Nazarenko[1,2], Alexander A. Prokhorov[1,2]
[1]DATADVANCE, Moscow, Russia
[2]Institute for information transmission problems RAS (Kharkevich Institute), Moscow, Russia

ABSTRACT

Simulation experiments performed while solving multidisciplinary engineering and scientific problems require joint usage of multiple software tools. Further, when following a preset plan of experiment or searching for optimum solutions, the same sequence of calculations is run multiple times with various simulation parameters, input data, or conditions while overall workflow does not change. Automation of simulations like these requires implementing of a workflow where tool execution and data exchange is usually controlled by a special type of software, an integration environment or platform. The result is an integration workflow (a platform-dependent implementation of some computing workflow) which, in the context of automation, is a composition of weakly coupled (in terms of communication intensity) typical subtasks. These compositions can then be decomposed back into a few workflow patterns (types of subtasks interaction). The patterns, in their turn, can be interpreted as higher level subtasks.

This paper considers execution control and data exchange rules that should be imposed by the integration environment in the case of an error encountered by some integrated software tool. An error is defined as any abnormal behavior of a tool that invalidates its result data thus disrupting the data flow within the integration workflow. The main requirement to the error handling mechanism implemented by the integration environment is to prevent abnormal termination of the entire workflow in case of missing intermediate results data. Error handling rules are formulated on the basic pattern level and on the level of a composite task that can combine several basic patterns as next level subtasks. The cases where work-

flow behavior may be different, depending on user's purposes, when an error takes place, and possible error handling options that can be specified by the user are also noted in the work.

Key words: error handling, process integration, process automation, workflow, workflow patterns.

REFERENCES

1. Prokhorov A.A., Nazarenko A.M., Davydov A.V. Patterns of engineering and scientific workflows. CEUR Workshop Proceedings, 2016, vol. 1787, pp. 403–409. (in Russian)

2. Radchenko G.I. *Grid-sistema CAEBeans: integraciya resursov inzhenernyh paketov v raspredelennye vychislitelnye sredy* [CAEBeans grid: CAE software integration for disctributed calculations]. Vestnik of Lobachevsky University of Nizhni Novgorod, 2009, no. 6 (1), pp. 192–202. (in Russian).

3. Deelman E., Gannon D., Shields M., Taylor I. Workflows and e-Science: An overview of workflow system features and capabilities. Future Generation Computer Systems, 2009, vol. 25, no. 5, pp. 528–540.

4. Burnaev E.V., Gubarev F.V., Morozov S.M., Prohkorov A.A., Khominich D.S. *Avtomatizaciya inzhenernyh raschetov analiz dannyh i optimizaciya s pomoshchyu programmnogo kompleksa PSE/MACROS* [Process integration, design optimization and data analysis with PSE/MACROS framework]. Interindustry information service, 2012, no. 4 (165), pp. 41–50. (in Russian)

5. Knyazkov K.V., Kovalchuk S.V., Tchurov T.N., Maryin S.V., Boukhanovsky A.V. CLAVIRE: e-Science infrastructure for data-driven computing. Journal of Computational Science, 2012, vol. 3, no. 6, pp. 504–510.

6. Sukhoroslov O., Volkov S., Afanasiev A. A Web-Based Platform for Publication and Distributed Execution of Computing Applications. 14th International Symposium on Parallel and Distributed Computing (ISPDC). IEEE, 2015, pp. 175–184.

7. Luo Z. et al. Exception handling in workflow systems. Applied Intelligence, 2000, vol. 13, no. 2, pp. 125–147.

8. Hagen C., Alonso G. Exception handling in workflow management systems. IEEE Transactions on software engineering, 2000, vol. 26, no. 10, pp. 943–958.

9. Russell N., van der Aalst W., ter Hofstede A. Workflow exception patterns. International Conference on Advanced Information Systems Engineering. Springer Berlin Heidelberg, 2006, pp. 288–302.

10. Nazarenko A.M., Prokhorov A.A. Hierarchical Dataflow Model with Automated File Management for Engineering and Scientific Applications. Procedia Computer Science, 2015, vol. 66, pp. 496–505.

11. Nazarenko A.M., Perestoronin N.O., Prokhorov A.A. *Upravlenie fajlami v ramkakh modeli potokov dannykh dlya raspredelennykh vychislenij* [File management in distributed dataflow-based workflows.]. Civil Aviation High Technologies, 2016, vol. 19, no. 05, pp. 161–172. (in Russian)

INFORMATION ABOUT THE AUTHORS

Alexey M. Nazarenko, Senior Programmer, DATADVANCE LLC; Junior Research Fellow, Institute of Information Transmission Problems of Russian Academy of Sciences (Kharkevich Institute), alexey.nazarenko@datadvance.net.

Alexander A. Prokhorov, Head of Software Division, DATADVANCE LLC; Research Fellow, Institute of Information Transmission Problems of Russian Academy of Sciences (Kharkevich Institute), alexander.prokhorov@datadvance.net.

Поступила в редакцию	06.03.2017	Received	06.03.2017
Принята в печать	27.04.2017	Accepted for publication	27.04.2017