УДК 004.942 DOI: 10.14529/mmph250404

МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ОЦЕНИВАНИЯ НАДЁЖНОСТИ ПРОГРАММНЫХ СРЕДСТВ ВЫЧИСЛИТЕЛЬНЫХ КЛАСТЕРОВ

А.Н. Привалов[1], А.В. Богомолов[2], Е.В. Ларкин[3], Т.А. Акименко[3]

[1] Тульский государственный педагогический университет им. Л.Н. Толстого, г. Тула, Российская Федерация

[2] Федеральный исследовательский центр «Информатика и управление» Российской академии наук, г. Москва, Российская Федерация

[3] Тульский государственный университет, г. Тула, Российская Федерация

E-mail: privalov.61@mail.ru

Аннотация. Представлены результаты синтеза математического обеспечения априорного оценивания надежности программных средств вычислительных кластеров, учитывающего специфику их построения: параллельная обработка информации, высокая производительность, масштабируемость, повышенная отказоустойчивость, балансировка нагрузки, поддержка гетерогенных конфигураций. Разработанное математическое обеспечение основано на линеаризации исходного последовательного алгоритма и преобразовании его из линейной формы в параллельную. Линеаризация исходной структуры программного средства позволяет представить реализуемый им алгоритм в виде объединения последовательностей операторов, начинающихся и заканчивающихся в неисполнимых операторах. Процедура линеаризации включает этапы представления алгоритма, описывающего последовательную обработку данных, в виде графа; линеаризацию графа путем формирования графа реализаций с помощью матричной конкатенации матрицы его смежности; формирование параллельной структуры из линеаризованной модели за счет разделения последовательности операторов на фрагменты, число которых равно числу вычислительных кластеров. Получены математические зависимости расчета оценок вероятностей появления последовательностей операторов, не удовлетворяющих исходным требованиям к алгоритму, на основании законов распределений обрабатываемых данных и условий ветвления вычислительного процесса. Рассмотрены модели отказов, причинами которых являются ошибки синхронизации обработки данных в вычислительных кластерах. Поиск и оценивание вероятностей появления на линейной и параллельной форме ветвей, не удовлетворяющих требованиям технического задания на программное средство, позволило построить модель генератора отказов программных средств, обеспечивающего априорное оценивание времени наработки до отказа. Моделируемый поток отказов рассматривается как стохастическая сумма пуассоновских потоков, формируемых при кратном циклическом запуске вычислительного кластера. Время наработки программного средства на отказ рассчитывается на основании плотности распределения времени между отказами в пуассоновском потоке. Синтезированы математические модели генератора отказов и расчета оценки времени наработки программных средств на отказ, основанные на оценках вероятностей попадания параллельного вычислительного процесса в отказные ветви реализуемых алгоритмов. Приоритеты развития полученных результатов связываются с разработкой моделей отказов программных средств вычислительных кластеров вследствие структурных ошибок, допущенных при разработке параллельных программ.

Ключевые слова: линеаризация графа; ветвление алгоритма; распараллеливание вычислений; наработка на отказ; надежность программного средства; программное средство вычислительного кластера.

Введение

Интеллектуализация и цифровизация промышленных технологий предполагает существенное повышение производительности средств обработки данных [1–3]. Это может быть достигну-

то за счет внедрения универсальных вычислительных кластеров, объединяющих множество независимых компьютеров-узлов в единую систему с общей файловой системой, высокоскоростной коммуникационной сетью и распределенной архитектурой [4, 5].

Широкое применение вычислительных кластеров актуализирует вопросы обеспечения надежности их аппаратных и программных средств. Надежность аппаратных средств обеспечивается за счет совершенствования технологий их проектирования, изготовления и эксплуатации, и в настоящее время является хорошо изученным вопросом [6, 7]. Программные средства вычислительных кластеров обладают рядом отличительных особенностей, выделяющих их среди других программных средств: параллельная обработка информации, высокая производительность, масштабируемость, повышенная отказоустойчивость, балансировка нагрузки, поддержка гетерогенных конфигураций – поэтому вопросы обеспечения их надежности требуют изучения [8–11].

Разработка технологий повышения надежности программных средств требует наличия модели формирования, выявления и рискометрии ошибок [12]. Структура модели должна формироваться по результатам анализа алгоритма функционирования вычислительного кластера, но не должна быть идентичной ему, поскольку ошибки программирования означают, что разработанное программное средство не соответствует заданным техническим требованиям, а модель должна выявлять именно эти различия, приводящие к отказам [13–15].

Основным недостатком известных подходов к оцениванию надежности программных средств [16, 17] является то, что:

– они ориентированы на исследование надежности программных средств однопроцессорных ЭВМ;

– оценки параметров надежности рассчитываются на основе эмпирических данных о количестве ошибок, выявленных между двумя наблюдениями за работой тестируемой программы, а следовательно, на результат оценивания сильно влияют как период наблюдения, так и покрытие области пространства обрабатываемых данных подобластью данных, генерируемых тестирующей программой.

Вместе с тем цена ошибки при анализе надежности вычислительных кластеров является, как правило, высокой, поэтому задача разработки надежной программы сводится к выявлению и устранению потенциальных причин отказов на этапе разработки, до ввода программных средств в эксплуатацию. Априорное оценивание надежности программных средств вычислительных кластеров на этапе их разработки недостаточно распространено в инженерной практике. Поэтому проведено исследование, имевшее целью разработку математического обеспечения априорного оценивания надёжности программных средств вычислительных кластеров.

Схема вычислительного процесса

Программные средства вычислительных кластеров в процессе разработки проходят следующие этапы [18], на каждом из которых возможно появление ошибок, обусловленных человеческим фактором.

Этап 1. Разработка алгоритма, описывающего последовательную обработку данных в виде графа (рис. 1, *а*), структура которого описывается выражением $G=\{A,\boldsymbol{r}\}$, где $A=\{a_0,a_1,...,a_j,...,a_{J+1}\}$ – множество вершин, каждая из которых является абстрактным аналогом оператора алгоритма; $\boldsymbol{r}=\left[r_{i,j}\right]$ – матрица смежности размерности $(J+2)(J+2)$, описывающая связи между операторами алгоритма; $r_{i,j}=(a_i,a_j)$.

В общем случае алгоритм включает неисполнимые операторы «Начало» (a_0) и «Конец» (a_{J+1}), а также *J* исполнимых операторов, представляемых вершинами $a_1...a_J$. Оператор «Начало» выполняет перезапуск алгоритма, а оператор «Конец» моделирует завершение обработки данных. Факт перезапуска алгоритма моделируется с помощью дуги (a_{J+1},a_0), показанной на рис. 1, *а* пунктирной стрелкой. В матрице смежности элемент $r_{i,j}=(a_i,a_j)$ обозначает дугу, ведущую из a_i в a_j; $1\le i,\ j\le J+1$. Значение $r_{i,j}=\varnothing$, где $\varnothing$ – пустой элемент, означает, что в структуре графа *G* дуга (a_i,a_j) отсутствует.

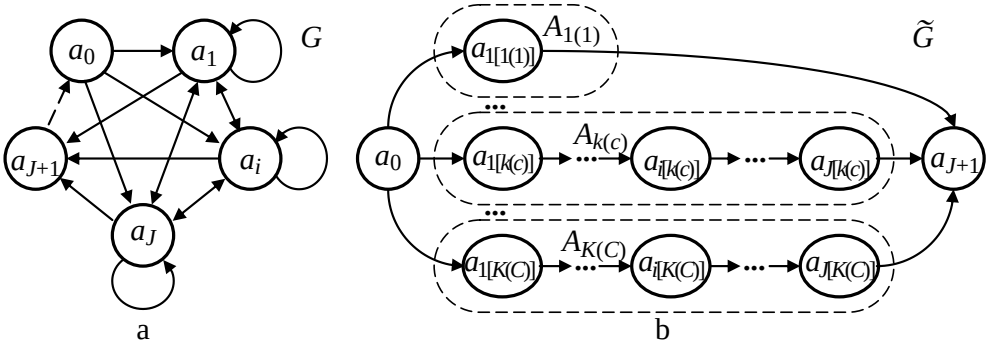

Рис. 1. Исходный последовательный алгоритм обработки данных в вычислительном кластере (a) и его линеаризованная версия (b)

Корректно сконструированный последовательный алгоритм обладает свойствами:

– в оператор «Начало» можно войти только при запуске программы, то есть $r_{i,0} = \varnothing, 0 \le i \le J+1$;

– попадание в оператор «Конец» означает остановку исполнения программы то есть $r_{J+1,j} = \varnothing, 0 \le j \le J+1$;

– среди исполнимых операторов отсутствуют операторы, образующие автономные и полуавтономные циклы первого и второго рода, то есть в структуре графа G существует хотя бы один путь, ведущий из a_0 в a_i, и хотя бы один путь, ведущий из a_i в a_{J+1}, $0 \le i \le J$;

– каждый путь, ведущий из a_0 в a_{J+1}, включает хотя бы одну вершину подмножества $\{a_1,...,a_J\}$.

Невыполнение любого из этих свойств означает, что при составлении последовательного алгоритма допущена ошибка, которая является источником потенциальных отказов программных средств вычислительного кластера.

Этап 2. Линеаризация графа *G*, для чего матрица его смежности $\boldsymbol{r}$ возводится в *c*-ю конкатенационную степень, при этом формируется граф реализаций (рис. 1, *б*)

$$\tilde{G} = \boldsymbol{I}_0 \,\&\left[\bigcup_{c(\&)=1}^{C(\&)} \boldsymbol{r}^{c(\&)}\right]\&\,\boldsymbol{I}_{J+1},$$

где $\boldsymbol{r}^{c(\&)}$ – операция возведения матрицы смежности в $c(\&)$-ю конкатенационную степень; $\&$ – обозначение логической операции конкатенации; $\boldsymbol{I}_{R,0} = (a_0, \varnothing, ..., \varnothing)$, $\boldsymbol{I}_{C,J+1} = (\varnothing, ..., \varnothing, a_{J+1})^{\theta}$ – соответственно вектор-строка размерности $1\times(J+2)$ и вектор-столбец размерности $(J+2)\times 1$; θ – обозначение операции транспонирования; $C(\&)$ – максимальное значение конкатенационной степени, в которую возводится матрица смежности $\boldsymbol{r}$; $\varnothing$ – пустой элемент.

Возведение матрицы смежности в $c(\&)$-ю конкатенационную степень выполняется в виде рекуррентной процедуры по правилам возведения алгебраической матрицы в степень, в которой операция перемножения членов заменена на операцию конкатенации, а операция сложения произведений – на операцию объединения конкатенаций, то есть

$$\boldsymbol{r}^{c(\&)} = \left[r_{i,j}^{c(\&)-1}\right]\&\left[r_{i,j}\right] = \bigcup_{l=0}^{J+1}\left(r_{i,l}^{c(\&)-1}\,\&\,r_{l,j}\right) = \left[r_{i,j}^{c(\&)}\right],$$

где $\&$ – матричная конкатенация; $\boldsymbol{r}^{c(\&)-1}$ – $\left[c(\&)-1\right]$-я конкатенационная степень матрицы $\boldsymbol{r}$; $r_{i,j}^{c(\&)-1}$ – элемент матрицы $\boldsymbol{r}^{c(\&)-1}$, находящийся на пересечении *i*-й строки и *j*-го столбца.

Конкатенация ранее построенной последовательности вершин с пустым элементом прерывает последовательность. Конкатенация $\boldsymbol{I}_{R,0}$ с матрицей смежности производится по правилам умножения вектора-строки на матрицу, а матрицы смежности с вектором $\boldsymbol{I}_{C,J+1}$ – по правилам умножения матрицы на вектор-столбец.

В общем случае граф *G* имеет вид, показанный на рис. 1, *б*. Вершины графа образуют линей-

ные (не содержащие циклов и ветвлений) последовательности $A_{k(c)}$ исполнимых операторов алгоритма, задействованных в $k(c)$-й реализации. Последовательности имеют вид

$$A_{k(c)} = \left[a_{v[k(c)]} \right],$$

где $A_{k(c)}$ – векторы, состоящие из вершин $a_{i[k(c)]} \in A$, описывающие линейные последовательности операторов; $1 \le v \le J$, $a_{i[k(c)]}$ – вершины, размещенные в k-й реализации алгоритма на месте $i[k(c)]$; $i[k(c)]$ – индекс-функция, обозначающая номер i-й вершины в k-й линейной последовательности, получающейся в результате возведения матрицы $\boldsymbol{r}$ в $c(\&)$ -ю конкатенационную степень; $J[k(c)] = c$ – количество вершин в последовательности, равное конкатенационной степени, в которую возведена матрица $\boldsymbol{r}$; $K(c) = J^c$ – максимальное количество последовательностей длиной c, формируемых в результате возведения матрицы $\boldsymbol{r}$ в $c(\&)$ -ю конкатенационную степень.

Очевидно, что данные, обрабатываемые алгоритмом, являются дискретными и случайными. Поэтому они могут быть представлены в виде множества векторов

$$\coprod_{m=1}^{M} D_m = \left\{ \left[D_{0(m)} \right], ..., \left[D_{n(m)} \right], ..., \left[D_{N(m)-1} \right] \right\} = \left\{ \mathrm{D}_{0(\Sigma)}, ..., \mathrm{D}_{n(\Sigma)}, ..., \mathrm{D}_{N(\Sigma)-1} \right\},$$

где $D_m = D_{0(m)}, ..., D_{n(m)}, ..., D_{N(m)-1}$ – множество дискретных значений обрабатываемой величины; $\mathrm{D}_{n(\Sigma)} = \left[D_{n(1)}, ..., D_{n(m)}, ..., D_{n(M)} \right]$ – обобщенный вектор обрабатываемых данных, описывающий соответствующую комбинацию; $n(\Sigma)$ – номер комбинации; $\coprod$ – групповое декартово произведение множеств; $N(\Sigma) = \prod_{m=1}^{M} N(m)$.

Фактор случайности значений дискретных данных, поступающих на обработку, описывается континуальной плотностью распределения, которая представляется в виде взвешенной суммы δ-функций Дирака

$$f(\boldsymbol{D}) = \sum_{n(\Sigma)=0(\Sigma)}^{N(\Sigma)-1} p_{n(\Sigma)} \delta(\boldsymbol{D} - \boldsymbol{D}_{n(\Sigma)}),$$

где $p_{n(\Sigma)}$ – вероятность появления комбинации $\boldsymbol{D}_{n(\Sigma)}$; $\delta(\boldsymbol{D} - \boldsymbol{D}_{n(\Sigma)})$ – векторная δ-функция Дирака:

$$\delta(\boldsymbol{D} - \boldsymbol{D}_{n(\Sigma)}) = \delta(D_1 - D_{n(1)}) \cdot ... \cdot \delta(D_m - D_{n(m)}) \cdot ... \cdot \delta(D_M - D_{n(M)}), \int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} \delta(\boldsymbol{D} - \boldsymbol{D}_{n(\Sigma)}) d\boldsymbol{D} = 1.$$

Если элементы вектора $\boldsymbol{D}$ некоррелированы между собой, то для каждого элемента может быть определена плотность распределения

$$f_m(D_m) = \sum_{n(m)=0}^{N(m)-1} p_{n(m)} \delta(D_m - D_{n(m)}).$$

А общая плотность распределения в этом случае равна произведению плотностей отдельных случайных величин, то есть

$$f(\boldsymbol{D}) = \prod_{m=1}^{M} f_m(D_m).$$

Ветвление вычислительного процесса G определяется логическими условиями переключения из текущей вершины $a_{i[k(c)]} \in A_{k(c)}$ в смежные с ней вершины из подмножества $\{a_i\}$, $1 \le i \le J+1$. В свою очередь, если $a_{i[k(c)]} = a_i$, где $a_{i[k(c)]}$ – вершина, принадлежащая последовательности, сформированной в результате конкатенации, а a_i – вершина, принадлежащая подмножеству, то логические условия присоединения к последовательности $A_{k(c)}$ вершины из подмножества $\{a_i\}$ определяются вектором отношений

$$\mathbf{\Phi} = \left[\varphi_{i,j}\left(\boldsymbol{D}\right) \le 0 \right],$$

где $\varphi_{i,j}\left(\boldsymbol{D}\right)$ – некоторая функция от данных дискретного пространства $\tilde{\boldsymbol{D}}$, $1 \le i \le J+1$.

Принимая во внимание, что в программе отсутствуют полуавтономные циклы, в том числе «зависания», одно из условий отношения вектора **Φ** всегда выполняется, события выполнения отношений являются несовместными и составляют полную группу. С использованием отношений вектора **Φ** может быть получена оценка вероятности того, что при следующем выборе ветви программы к последовательности $A_{k(c)}$ будет добавлена вершина a_j, то есть $a_{i[k(c)]+1} = a_j$ будет равна

$$P\{i[k(c)] = i, i[k(c)]+1 = j\} = \int \ldots \int_{\varphi_{i,j}(\boldsymbol{D})<0} \ldots \int f(\boldsymbol{D}) d\boldsymbol{D}.$$

Вероятность появления последовательности $A_{k(c)}$ определяется как произведение вероятностей:

$$P[A_{k(c)}] = \prod_{i[k(c)]=1[k(c)]}^{J[k(c)]} P\{i[k(c)] = i, i[k(c)]+1 = j\}.$$

Преобразование $\tilde{G}$ графа *G* является эквивалентным, поэтому можно считать, что ошибок на этом этапе не возникает. Однако такое преобразование позволяет выявлять следующие ошибки в исходном алгоритме:

– наличие в структуре $\tilde{G}$ последовательностей с «неправильной» (большей или меньшей длины, определенной решаемой с помощью алгоритма задачи) длиной;

– выпадение из последовательностей структуры $\tilde{G}$ операторов, необходимых для решения задачи;

– наличие в последовательностях структуры $\tilde{G}$ «неправильных» и/или «лишних» операторов;

– наличие в последовательностях со структурой $\tilde{G}$ фрагментов с неправильным чередованием операторов, например, нарушающим отношение предшествования.

Для указанных последовательностей могут быть определены вероятности их появления в структуре $\tilde{G}$, а также общая вероятность отказа/сбоя программных средств при реализации алгоритма со структурой *G*:

$$p_{fail} = \sum_{\overline{k(c)}} P[\overline{k(c)}],$$

где $\overline{k(c)}$ – номера сбойных ветвей структуры $\tilde{G}$; $P[\overline{k(c)}]$ – вероятности появления сбойных ветвей в структуре алгоритма, оцениваемые как вероятности появления последовательностей $A_{k(c)}$.

Этап 3. Формирование параллельной структуры из линеаризованной модели $\tilde{G}$.

Один из способов формирования параллельной структуры основан на разделении последовательности $A_{k(c)}$ на *L* фрагментов, реализуемых на *L* вычислительных кластерах (рис. 2, *а*).

Модель представляет собой сеть Петри

$$\Pi = \left\{ \{\Pi_l\}, \{a_0, a_{J+1}\}, \{z_0, z_{J+1}\}, \Psi_{\text{in}}\{z_0, z_{J+1}\}, \Psi_{\text{out}}\{z_0, z_{J+1}\} \right\},$$

где Π_l – подсеть, описывающая последовательность операторов, исполняемых *l*-м вычислительным кластером, $1 \le l \le L$; $\{a_0, a_{J+1}\}$ – позиции, моделирующие неисполнимые операторы алгоритма; z_0, z_{J+1} – переходы, моделирующие соответственно начало и окончание параллельной обработки данных; $\Psi_{in}\{z_0, z_{J+1}\}$, $\Psi_{out}\{z_0, z_{J+1}\}$ – входная и выходная функция переходов; $\Psi_{in}(z_0) = a_0$; $\Psi_{out}(z_{J+1}) = a_{J+1}$; $\Psi_{in}(z_{J+1}) = \{\Pi_l\}$; $\Psi_{out}(z_0) = \{\Pi_l\}$.

В свою очередь, подсети Π_l, описывающие вычислительные кластеры (в индексе функции $l[k(c)]$ для краткости параметры $k(c)$ опущены), имеют вид

$$\Pi_l = \left\{ \left\{ a_{i(k,l)} \right\}, \left\{ z_{i(k,l)} \right\}, \Psi_{in} \left\{ z_{i(k,l)} \right\}, \Psi_{out} \left\{ z_{i(k,l)} \right\} \right\},$$

где $\{a_{i(k,l)}\}$ – множество позиций, $a_{i(k,l)} \in A_{k(c)}$; $\{z_{i(k,l)}\}$ – множество переходов; $\Psi_{in}\left[z_{i(k,l)} \right] = a_{i(k,l)}$; $\Psi_{out}\left[z_{i(k,l)} \right] = a_{i(k,l)} + 1$, $1(k,l) \le i(k,l) \le J(k,l) - 1$.

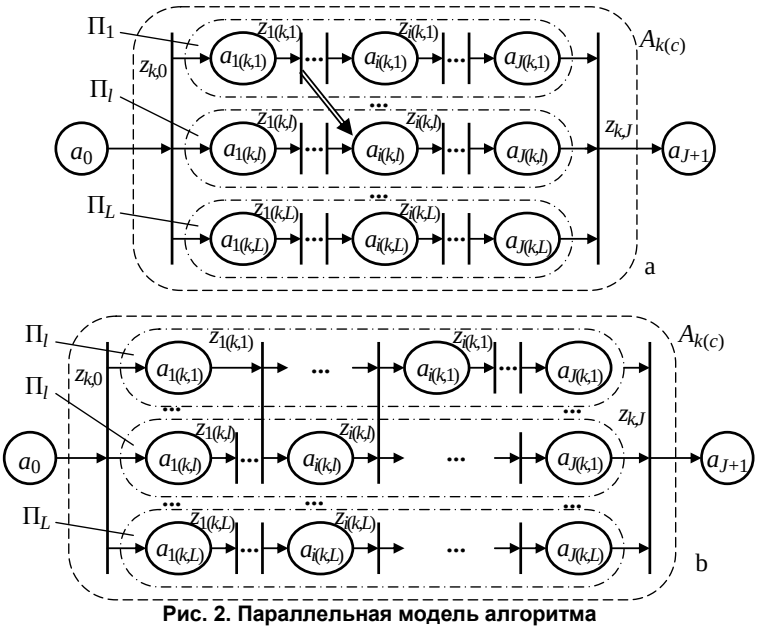

Рис. 2. Параллельная модель алгоритма

Сеть Петри описывает передачу управления от одного оператора к другому внутри вычислительных кластеров. Наряду с передачей управления внутри вычислительных кластеров естественным образом организуется передача данных, формируемых предшествующим оператором, всем последующим операторам. Простое разделение последовательности на фрагменты может породить ситуацию, когда данные, формируемые оператором *m*-го фрагмента, должны быть использованы операторами *l*-го фрагмента, а передача данных из *m*-го в *l*-й вычислительный кластер не предусмотрена (случай при *m* = 1 показан на рис. 2, *a* двойной стрелкой). Эта типичная ошибка распараллеливания может привести к отказу при реализации последовательности $A_{k(c)}$ в целом, а следовательно, вероятность такого события входит в формулу для расчета p_{fail} как одно из слагаемых.

Ошибки подобного типа устраняются программно за счет введения промежуточной синхронизации вычислительного процесса, проводимой на переходе $z_{1(k,1)}$ (рис. 2, *b*). При промежуточной синхронизации

$$\Psi_{in}[z_{1(k,1)}] = \{a_{1(k,1)}, a_{i(k,l)-1}\}, \ \Psi_{out}[z_{1(k,1)}] = \{a_{2(k,1)}, a_{i(k,l)}\}.$$

Такое введение входной и выходной функции на переходе $z_{1(k,1)}$ замедляет процесс кластерной обработки данных, но уменьшает вероятность отказов программных средств.

Генерация потока отказов

Модель генерации потока отказов в вычислительном кластере приведена на рис. 3. Алгоритм генерации представлен в виде графа

$$\Gamma = \{A, \boldsymbol{\rho}\},$$

где $A = \{\alpha_0, \alpha_1, \alpha_2\}$ – множество вершин; α_0 – моделирует неисполнимый оператор «Начало»; α_1 – моделирует исполнимый оператор обработки данных в вычислительном кластере; α_2 – моделирует неисполнимый оператор «Конец»; $\boldsymbol{\rho} = [\rho_{i,j}]$ – матрица смежности размерности 3×3:

$$\rho = \begin{bmatrix} \varnothing & \rho_{0,1} & \varnothing \\ \varnothing & \rho_{1,1} & \rho_{1,2} \\ \varnothing & \varnothing & \varnothing \end{bmatrix} = \begin{bmatrix} \varnothing & (\alpha_0,\alpha_1) & \varnothing \\ \varnothing & (\alpha_1,\alpha_1) & (\alpha_1,\alpha_2) \\ \varnothing & \varnothing & \varnothing \end{bmatrix}.$$

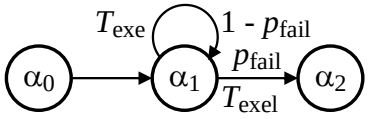

Рис. 3. Модель генерации потока отказов вычислительного кластера

При исполнении алгоритма в вычислительном кластере отказ происходит с вероятностью p_{fail} и обработка данных прекращается. С вероятностью $(1–p_{\text{fail}})$ программа завершается и перезапускается в штатном режиме. Исполнение оператора α_1 осуществляется за время, равное T_{exe} секунд.

Применение к матрице смежности $\boldsymbol{\rho}$ графа $\tilde{G}$, в котором $\boldsymbol{I}_0 = (a_0,\varnothing,\varnothing)$, $\boldsymbol{I}_2 = (\varnothing,\varnothing,\alpha_2)^\theta$ дает последовательности исполнимых операторов, сведенные в таблицу.

Формирование последовательностей исполнимых операторов

c	Последовательности операторов	Вероятность появления	Время до отказа
1	–	–	–
2	a_0, a_1, a_2	p_{fail}	T_{exe}
3	a_0, a_1, a_1, a_2	p_{fail}^2	$2T_{exe}$
	...		
k	$a_0, a_1, ..., a_1, ..., a_1, ...a_2$ $\lvert \quad c-1 \quad \rvert$	p_{fail}^{c-1}	$(c-1)T_{exe}$
	...		
K	$a_0, a_1, ..., a_1, ..., a_1, ...a_2$ $\lvert \quad C-1 \quad \rvert$	p_{fail}^{C-1}	$(C-1)T_{exe}$

Из таблицы следует, что поток отказов может рассматриваться как стохастическая сумма потоков, формируемых при одно-, двух-, трех- и более кратном циклическом запуске вычислительного кластера. В соответствии с теоремой Б.И. Григелиониса, такой поток стремится к пуассоновскому [19]. Плотность распределения времени между отказами в пуассоновском потоке равна [20–22]

$$f_{bf}(t) \cong T_{bf}^{-1} e^{-tT_{bf}^{-1}},$$

где T_{bf} – время наработки программного средства на отказ, оценка которого в предельном случае при $C \to \infty$ рассчитывается по зависимости

$$T_{bf} = T_{exe} p_{fail}.$$

Отметим, что при эксплуатации программных средств вычислительных кластеров ошибки, приводящие к отказам, как правило, устраняются [23–25]. Это означает, что количество сбойных ветвей в графе $\tilde{G}$ уменьшается, а, следовательно, и уменьшается вероятность p_{fail} – время наработки системы до отказа при этом увеличивается.

Заключение

Таким образом, разработано математическое обеспечение оценивания надежности программных средств вычислительных кластеров, основанное на линеаризации исходного последовательного алгоритма и преобразовании его из линейной формы в параллельную. Поиск и оценивание вероятностей появления на линейной и параллельной форме ветвей, не удовлетворяющих требованиям технического задания на программное средство, позволило построить модель генератора отказов программных средств и обеспечить априорное оценивание времени наработки до

отказа. Дальнейшее развитие исследований связывается с разработкой моделей отказов программных средств вычислительных кластеров вследствие структурных ошибок, допущенных при разработке параллельных программ.

Работа выполнена при поддержке гранта РНФ 25-29-20177.

Литература

1. Fadali, M.S. Digital Control Engineering: Analysis and Design / M.S. Fadali, A. Visioli. – Amsterdam, Elsevier Inc., 2013. – 552 p.

2. Larkin, E. Digital Control of Continuous Production with Dry Friction at Actuators / E. Larkin, A. Privalov, A. Bogomolov, T. Akimenko // Smart Innovation, Systems and Technologies. – 2022. – Vol. 232. – P. 427–436.

3. Larkin, E. Digital Control by Robot Manipulator with Improved Rigidity / E. Larkin, A. Bogomolov, A. Privalov // Smart Innovation, Systems and Technologies. – 2023. – Vol. 329. – P. 45–59.

4. Гримм, Р. Параллельное программирование на современном языке C++ / Р. Гримм. – М.: ДМК Пресс, 2022. – 616 с.

5. Chapman, B. Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation) / B. Chapman, G. Jost, R. Van der Pas. – Cambridge, Massachusetts: The MIT Press., 2008. – 353 p.

6. Naes, A. System Reliability Analysis by Enhanced Monte Carlo Simulation / A. Naes, B.J. Leira, O. Batsevych // Structural Safety. – 2009. – Vol. 31. – P. 349–355.

7. Sánchez-Silva, M. Reliability and Life-Cycle Analysis of Deteriorating Systems / M. Sánchez-Silva, G.-A. Klutke. – Switzerland: Springer, 2016. – 355 p.

8. Shatnavi, O. Measuring Commercial Software Operational Reliability. An Interdisciplinary Modelling Approach / O. Shatnavi // Eksploatacia i Niezawodnose – Maintenance and reliability. – 2014. – Vol. 16, no. 4. – P. 585–594.

9. Богомолов, А.В. Математические модели программных отказов цифровых систем управления / А.В. Богомолов, Е.В. Ларкин, А.Н. Привалов // Научно-техническая информация. Серия 2: Информационные процессы и системы. – 2025. – № 1. – С. 1–9.

10. Привалов, А.Н. Моделирование надежности программных компонентов киберфизических систем / А.Н. Привалов, А.В. Богомолов, Е.В. Ларкин // Программные продукты и системы. – 2025. – № 1. – С. 47–54.

11. Tobin, D. Model of Organization of Software Testing for Cyber-Physical Systems / D. Tobin, A. Bogomolov, M. Golosovskiy // Cyber-Physical Systems: Modelling and Industrial Application. Cham: Springer, 2022. – P. 51–60.

12. Анализ характера изменений программ и поиск неисправленных фрагментов кода / М.С. Арутюнян, Г.С. Иванов, В.Г.Варданян и др. // Труды Института системного программирования РАН. – 2019. – Т. 31, № 1. – С. 49–58.

13. Молчанов А.С. Испытания интеллектуальных систем и комплексов воздушной разведки с использованием цифровых портретов типовых объектов воздушной разведки / А.С. Молчанов, В.А. Коломоец // Наукоемкие технологии. – 2025. – Т. 26, № 3. – С. 20–31.

14. Кулямин, В.В. Конструирование программных систем, нацеленное на обеспечение безопасности / В.В. Кулямин, А.К. Петренко, Е.А. Рудина // Труды Института системного программирования РАН. – 2024. – Т. 36, № 5. – С. 7–16.

15. Математическое обеспечение иерархического цифрового управления сложным технологическим объектом / Е.В. Ларкин, А.В. Богомолов, А.Н. Привалов, Т.А. Акименко // Вестник Южно-Уральского государственного университета. Серия «Математика. Механика. Физика». 2024. – Т. 16, № 4. – С. 43–55.

16. Barack O. Assessment and Prediction of Software Reliability in Mobile Applications / O. Barack, L. Huang // Journal of Software Engineering and Applications. – 2020. – Vol. 13, no. 9. – P. 179–190.

17. Ларкин, Е.В. Математическое моделирование отказов программного обеспечения / Е.В. Ларкин, А.В. Богомолов, А.Н. Привалов // Вестник ЮУрГУ. Серия Математическое моделирование и программирование». – 2025. – Т. 18, № 3. – С. 73–86.

18. Reliability of Robot's Controller Software / E. Larkin, T. Akimenko, A. Bogomolov, V. Sharov // Lecture Notes in Computer Science. – 2023. – Vol. 14214. – P. 289–299.

19. Григелионис, Б.И. О сходимости сумм ступенчатых случайных процессов к пуассоновскому / Б.И. Григелионис // Теория вероятностей и ее применения. – 1963. – Т. 8, № 2. – С. 189–194.

20. Bielecki, T.R. Conditional Markov Chains: Properties, Construction and Structured Dependence / T.R. Bielecki, J. Jakubowski, M. Niewęgłowski // Stochastic Processes and their Applications. – 2017. – Vol. 127, no. 4. – P. 1125–1170.

21. Lu, H. A Functional Central Limit Theorem for Markov Additive Arrival Processes and Its Applications to Queueing Systems / H. Lu, G. Pang, M. Mandjes // Queueing Systems. – 2016. – Vol. 84, no. 3. – P. 381–406.

22. Kobayashi, H. Probability, Random Processes and Statistical Analysis / H. Kobayashi, B.L. Marl, W. Turin. – Cambridge, University Press, 2012. – 812 p.

23. Dubrova, E. Fault-Tolerant Design / E. Dubrova. NY: Springer Science+Business Media, 2013. – 185 p.

24. Сенсорные сети контроля состояния авиационной техники при испытаниях и эксплуатации / Е.С. Солдатов, А.В. Богомолов, Е.В. Ларкин, А.С. Солдатов // Авиакосмическое приборостроение. – 2024. – № 2. – С. 61–68.

25. Голосовский, М.С. Алгоритм настройки систем нечёткого логического вывода на основе статистических данных / М.С. Голосовский, А.В. Богомолов, Д.С. Тобин // Научно-техническая информация. Серия 2: Информационные процессы и системы. – 2023. – № 1. – С. 1–9.

Поступила в редакцию 18 сентября 2025 г.

Сведения об авторах

Привалов Александр Николаевич – доктор технических наук, профессор, директор института передовых информационных технологий, Тульский государственный педагогический университет им. Л.Н. Толстого, г. Тула, Российская Федерация, e-mail: privalov.61@mail.ru.

Богомолов Алексей Валерьевич – доктор технических наук, профессор, ведущий научный сотрудник, Федеральный исследовательский центр «Информатика и управление» РАН, г. Москва, Российская Федерация, e-mail: a.v.bogomolov@gmail.com.

Ларкин Евгений Васильевич – доктор технических наук, профессор, профессор-консультант, Тульский государственный университет, г. Тула, Российская Федерация, e-mail: elarkin@mail.ru

Акименко Татьяна Алексеевна – кандидат технических наук, доцент, Тульский государственный университет, г. Тула, Российская Федерация, e-mail: tantan72@mail.ru.

Bulletin of the South Ural State University
Series "Mathematics. Mechanics. Physics"
2025, vol. 17, no. 4, pp. 24–34

DOI: 10.14529/mmph250404

MATHEMATICAL SUPPORT FOR ASSESSING THE RELIABILITY OF SOFTWARE FOR COMPUTING CLUSTERS

A.N. Privalov[1], A.V. Bogomolov[2], E.V. Larkin[3], T.A. Akimenko[3]
[1] Tolstoy Tula State Pedagogical University, Tula, Russian Federation
[2] Federal Research Center "Informatics and Control" of the Russian Academy of Sciences, Moscow, Russian Federation
[3] Tula State University, Tula, Russian Federation
E-mail: privalov.61@mail.ru

Abstract. The article presents the results of the synthesis of mathematical support for an a priori reliability assessment of software for computing clusters, taking into account the specifics of their design: parallel information processing, high performance, scalability, increased fault tolerance, load balancing,

and support for heterogeneous configurations. The developed mathematical support is based on the linearization of the original sequential algorithm and its transformation from a linear form to a parallel one. Linearization of the original structure of the software allows representing the algorithm it implements as a union of sequences of operators that begin and end in non-executable operators. The linearization procedure includes the following stages: representing the algorithm describing sequential data processing as a graph; linearization of the graph by forming a graph of implementations using matrix concatenation of its adjacency matrix; formation of a parallel structure from the linearized model by dividing the sequence of operators into fragments, the number of which is equal to the number of computing clusters. The paper presents the mathematical dependencies for calculating probability estimates for the occurrence of operator sequences that do not satisfy the initial requirements for the algorithm, based on the distribution laws of the processed data and the branching conditions of the computational process. It considers failure models caused by data processing synchronization errors in computing clusters. The study also involved searching for and estimating the probabilities of branches occurring in linear and parallel forms that do not meet the requirements of the software tool's technical specifications. As a result, we constructed a model of a software failure generator that provides an a priori estimate of the mean time to failure. The simulated failure flow is considered as a stochastic sum of Poisson flows formed during multiple cyclic launches of the computing cluster. The mean time between failures of the software is calculated based on the density of the distribution of the time between failures in the Poisson flow. Mathematical models of the failure generator and the calculation of the mean time between failures are synthesized, based on estimates of the probabilities of the parallel computing process entering the failure branches of the implemented algorithms. The priorities for the development of the obtained results are associated with the development of models of software failures of computing clusters due to structural errors made during the development of parallel programs.

Keywords: graph linearization; algorithm branching; parallel computing; mean time between failures; software reliability; reliability modeling; computing cluster software.

References

1. Fadali M.S., Visioli A. *Digital Control Engineering: Analysis and Design*. Amsterdam: Elsevier Inc, 2013, 552 p.

2. Larkin E., Privalov A., Bogomolov A., Akimenko T. Digital Control of Continuous Production with Dry Friction at Actuators. *Smart Innovation, Systems and Technologies*, 2022, Vol. 232, pp. 427–436.

3. Larkin E., Bogomolov A., Privalov A. Digital Control by Robot Manipulator with Improved Rigidity. *Smart Innovation, Systems and Technologies*, 2023, Vol. 329, pp. 45–59.

4. Grimm R. *Parallel Programming in the Modern C++ Language*. Moscow: DMK Press, 2022. 616 p.

5. Chapman B., Jost G., Van der Pas R. *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*. Cambridge, Massachusetts, The MIT Press., 2008, 353 p.

6. Naes A., Leira B.J., Batsevych O., System Reliability Analysis by Enhanced Monte Carlo Simulation. *Structural Safety*, 2009, Vol. 31, pp. 349–355.

7. Sánchez-Silva M., Klutke G.-A. *Reliability and Life-Cycle Analysis of Deteriorating Systems*. Switzerland: Springer, 2016. 355 p.

8. Shatnavi O. Measuring Commercial Software Operational Reliability. An Interdisciplinary Modeling Approach. *Eksploatacia i Niezawodnose- Maintenance and reliability*, 2014, Vol. 16, no. 4, pp. 585–594.

9. Bogomolov A.V., Larkin E.V., Privalov A.N. Mathematical Models of Software Failures of Digital Control Systems. *Scientific and Technical Information. Series 2: Information Processes and Systems*, 2025, no. 1, pp. 1–9. (in Russ.).

10. Privalov A.N., Bogomolov A.V., Larkin E.V. Reliability Modeling of Software Components of Cyber-Physical Systems. *Software Products and Systems*, 2025, no. 1, pp. 47–54. (in Russ.).

11. Tobin D., Bogomolov A., Golosovskiy M. Model of Organization of Software Testing for Cyber-Physical Systems. *Cyber-Physical Systems: Modeling and Industrial Application*. Cham: Springer, 2022, pp. 51–60.

12. Arutyunyan M.S., Ivanov G.S., Vardanyan V.G., Aslanyan A.K., Avetisyan A.I., Kurmangaleev Sh.F. Analysis of the Nature of Program Changes and Search for Uncorrected Code Fragments. *Proceedings of the Institute for System Programming of the Russian Academy of Sciences*, 2019, Vol. 31, no. 1, pp. 49–58. (in Russ.).

13. Molchanov A.S., Kolomoets V.A. Testing Intelligent Systems and Aerial Reconnaissance Complexes using Digital Portraits of Typical Aerial Reconnaissance Objects. *Science-Intensive Technologies*, 2025, Vol. 26, no. 3, pp. 20–31. (in Russ.).

14. Kulyamin V.V., Petrenko A.K., Rudina E.A. Design of Software Systems Aimed at Ensuring Security. *Proceedings of the Institute for System Programming of the Russian Academy of Sciences*, 2024, Vol. 36, no. 5, pp. 7–16. (in Russ.).

15. Larkin E.V., Bogomolov A.V., Privalov A.N., Akimenko T.A. Mathematical Support for Hierarchical Digital Control of a Complex Technological Object. *Bulletin of the South Ural State University. Series: Mathematics. Mechanics. Physics*, 2024, Vol. 16, no. 4, pp. 43–55. (in Russ.). DOI: 10.14529/mmph240406

16. Barack O., Huang L. Assessment and Prediction of Software Reliability in Mobile Applications. *Journal of Software Engineering and Applications*, 2020, Vol. 13, no. 9, pp. 179–190.

17. Larkin E.V., Bogomolov A.V., Privalov A.N. Mathematical Modeling of Software Failures. *Bulletin of the South Ural State University. Series: Mathematical Modeling and Programming*, 2025, Vol. 18, no. 3, pp. 73–86. (in Russ.).

18. Larkin E., Akimenko T., Bogomolov A., Sharov V. Reliability of Robot's Controller Software. *Lecture Notes in Computer Science*, 2023, Vol. 14214, pp. 289–299.

19. Grigelionis B.I. On the Convergence of Sums of Step Random Processes to a Poisson Process. *Probability Theory and Its Applications*, 1963, Vol. 8, no. 2, pp. 189–194.

20. Bielecki, T.R., Jakubowski, J., Niewęgłowski, M. Conditional Markov Chains: Properties, Construction, and Structured Dependence. *Stochastic Processes and Their Applications*, 2017, Vol. 127, no. 4, pp. 1125–1170.

21. Lu H., Pang G., Mandjes M. A Functional Central Limit Theorem for Markov Additive Arrival Processes and its Applications to Queuing Systems. *Queuing Systems*, 2016, Vol. 84, no. 3, pp. 381–406.

22. Kobayashi H., Marl B.L., Turin W. *Probability, Random processes and statistical analysis*. Cambridge: University Press, 2012, 812 p.

23. Dubrova E. *Fault-Tolerant Design*. NY: Springer Science+Business Media, 2013, 185 p.

24. Soldatov E.S., Bogomolov A.V., Larkin E.V., Soldatov A.S. Sensor networks for Monitoring the Condition of Aircraft Equipment during Testing and Operation. *Aerospace instrumentation*, 2024, no. 2, pp. 61–68. (in Russ.).

25. Golosovsky M.S., Bogomolov A.V., Tobin D.S. Algorithm for Tuning Fuzzy Logical Inference Systems Based on Statistical Data. *Scientific and technical information. Series 2: Information processes and systems*, 2023, no. 1, pp. 1–9. (in Russ.).

Received September 18, 2025

Information about the authors

Privalov Aleksandr Nikolaevich is Dr. Sc. (Engineering), Professor, Director of the Institute of Advanced Information Technologies of Tolstoy Tula State Pedagogical University, Tula, Russian Federation, e-mail: privalov.61@mail.ru.

Bogomolov Aleksey Valerievich is Dr. Sc. (Engineering), Professor, Leading Researcher of the Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Moscow, Russian Federation, e-mail: a.v.bogomolov@gmail.com.

Larkin Eugeny Vasilyevich is Dr. Sc. (Engineering), Professor, Professor-Consultant of Tula State University, Tula, Russian Federation, e-mail: elarkin@mail.ru.

Akimenko Tatyana Alekseevna is Cand. Sc. (Engineering), Associate Professor, Associate Professor of Tula State University, Tula, Russian Federation, e-mail: tantan72@mail.ru.