УДК 681.32

ОПТИМИСТИЧЕСКИЙ АЛГОРИТМ СИНХРОНИЗАЦИИ С ВНУТРИКОМПОНЕНТНЫМИ ОТКАТАМИ ДЛЯ РАСПРЕДЕЛЕННОГО МОДЕЛИРОВАНИЯ

Мирецкий А.В., Ладыженский Ю.В.
Донецкий Национальный Технический Университет

В статье представлен новый алгоритм отката для оптимистических алгоритмов синхронизации при распределенном дискретно-событийном моделировании систем. Описанный алгоритм основан на понятии компонентов. Он позволяет ускорить моделирование за счет уменьшения глубины выполняемых откатов, вплоть до отката состояния и отмены событий только одного компонента моделируемой системы.

Введение. Основная идея оптимистического алгоритма Time Warp заключается в том, что логические процессы обрабатывают события, «оптимистически» предполагая, что в результате этой обработки не возникнет ошибок причинности [1]. Если в процессе моделирования логический процесс принимает от другого процесса событие, которое должно произойти в прошлом, то выполняется возврат состояния моделируемой системы до этого момента в прошлом. Основные недостатки алгоритма Time Warp: 1) большие временные затраты на выполнение откатов; 2) Time Warp использует большие объемы памяти для хранения истории изменения состояния системы, необходимой для откатов**.**

Цель данной работы – снижение временных затрат на выполнение откатов за счет уменьшения глубины выполняемых откатов.

Локальные откаты. Одним из основных недостатков откатов в алгоритме Time Warp является отмена всех событий в рамках одного или нескольких логических процессов. По завершении отката, отмененные события должны быть выполнены заново. Чем глубже был откат, тем больше событий должно быть выполнено для достижения логическими процессами тех отметок модельного времени, которые были достигнуты до отката. Это может привести к значительным задержкам процесса моделирования.

Уже разработан ряд алгоритмов, позволяющих снизить временные издержки на выполнение откатов. Однако во всех

алгоритмах откат производится в одном или нескольких логических процессах в целом.

В [2] описан алгоритм SRADS with Local Rollbacks (Агрессивная обработка с локальными откатами), который позволяет полностью избавиться от анти-сообщений, инициализирующих откаты в соседних логических процессах. Т.е. все откаты будут локальными, внутри одного логического процесса. Каждый логический процесс выполняет агрессивную обработку событий (как в Time Warp), но результаты этой обработки (новые сообщения) не передаются другим логическим процессам до тех пор, пока не будет гарантий, что они никогда не будут отменены. Сообщение никогда не будет отменено, если его временная метка не превышает глобальное виртуальное время.

Внутрикомпонентные откаты. Предлагаемый алгоритм основан на алгоритме локальных откатов, и позволяет снизить глубину этих локальных откатов.

Известно, что многие сложные системы можно представить множеством более простых элементов (подсистем), взаимодействующих между собой. Такие элементы назовем компонентами. В терминах дискретно-событийного моделирования, компоненты способны обрабатывать и, при необходимости, порождать новые события.

Таким образом, модель представляется набором компонентов, связанных между собой определенным образом. Связанные компоненты могут обмениваться между собой событиями. Формально, модель представляется графом $M=(C, L)$, где $C=\{c\}$ – множество компонентов модели; $L=\{l \mid l=(c_k, c_m), c_k, c_m \in C, c_k \neq c_m\}$ – множество связей между компонентами.

Физическая система представляется множеством физических процессов, взаимодействующих между собой. Каждый физический процесс моделируется логическим процессом, а взаимодействие между физическими процессами моделируется обменом сообщениями, имеющими временные метки, между соответствующими логическими процессами. [1]

Каждый логический процесс выполняется на отдельном процессоре.

Пусть $LP = \{LP_i\}$, $|LP|=N$ – множество логических процессов, выполняемых на N процессорах. Модель М разделена на подмодели, моделируемые логическими процессами:

$$S = \left\{ s_i \mid s_i = (C_i, L_i), s_i \subseteq M, s_i \cap s_j = \varnothing \ \forall i, j : i \neq j, \bigcup_i s_i \cup (\varnothing, L_{bound}) = M \right\},$$

где s_i – подмодель, моделируемая логическим процессом LP_i; $C_i \subseteq C$ – множество компонентов подмодели s_i; $L_i \subseteq L$ – множество связей между компонентами подмодели s_i; $L_{bound} \subseteq L$, $L_{bound} = \{l_{bound} \mid l_{bound} = (c_k, c_m), c_k \in s_i, c_m \in s_i, i \neq j\}$ – множество связей между компонентами, находящимися в разных подмоделях. |S|=N.

Каждый логический процесс LP_i характеризуется локальным модельным временем T_i, которое представляет собой модельное время, достигнутое LP_i в процессе моделирования. В общем случае локальные модельные времена процессов отличаются друг от друга: $T_i \neq T_j, i \neq j$.

Пусть $B = \{b \mid b \in C, \exists\, l=(c_k, c_m): (b \equiv c_k \text{ или } b \equiv c_m) \text{ и } l \in L_{bound}\}$ – множество пограничных компонентов, т.е. таких компонентов, которые способны обмениваться сообщениями с другими пограничными компонентами, моделируемыми другими логическими процессами.

Пусть e – событие; $e = (\tau_e, \sigma_e, c_e, h_e)$, где τ_e – временная метка, показывающая, когда должно произойти событие; σ_e – временная метка, показывающая, когда было порождено событие; c_e – компонент, породивший событие; h_e – компонент, который должен обработать событие. $\sigma_e \leq \tau_e \ \forall\ e$, т.е. событие не может быть порождено позже, чем будет обработано.

Любое событие может быть следствием только обработки другого события. Событие не может быть порождено самостоятельно, т.е. между событиями имеются причинно-следственные связи. Если $e_1 = (\tau_1, \sigma_1, c_1, h_1)$ – событие-причина, а $e_2 = (\tau_2, \sigma_2, c_2, h_2)$ – событие-следствие, то $h_1 = c_2 \cup \tau_1 = \sigma_2$, т.е. событие-следствие порождается в тот же момент времени, когда происходит обработка события-причины, и событие-следствие порождается тем же компонентом, который обрабатывает событие-причину.

Каждый логический процесс моделируется последовательно на одном процессоре. Пусть LP_1, LP_2 – некоторые логические процессы, C_1, C_2 – множества компонентов моделируемых, соответствующими процессами, B_1, B_2 – пограничные компоненты соответствующих логических процессов. Пусть компонент $c' \in B_1$ порождает некоторое событие $e=(\tau, \sigma, c, h)$, тогда $\sigma = T_1$, $c = c'$, $\tau \geq \sigma \rightarrow \tau \geq T_1$. Если $h \in B_2$, то метка τ может быть любой по отношению к T_2. Если $h \in B_2$ и $\tau < T_2$, то событие e называется отставшим событием.

Если логический процесс принимает отставшее событие, то это событие нарушает ограничение причинности [1]. Логический процесс должен восстановить причинно-следственные связи между событиями.

Рассмотрим процесс отката при получении процессом LP_2 отставшего события из LP_1 (см. рис. 1). Пусть $C_2 = \{c_1, c_2, c_3, c_4\}$, $B_2=\{c_1\}$. Компонент c_1 получает отставшее событие $e_s=(\tau_s, \sigma_s, c_s, h_s)$, где $h_s = c_1$ *и* $\tau_s < T_2$. Алгоритм Time Warp предусматривает отмену всех событий $e_i=(\tau_i, \sigma_i, c_i, h_i) \mid \forall i: \tau_i > \tau_s$, и откат состояния всего логического процесса до отметки τ_s.

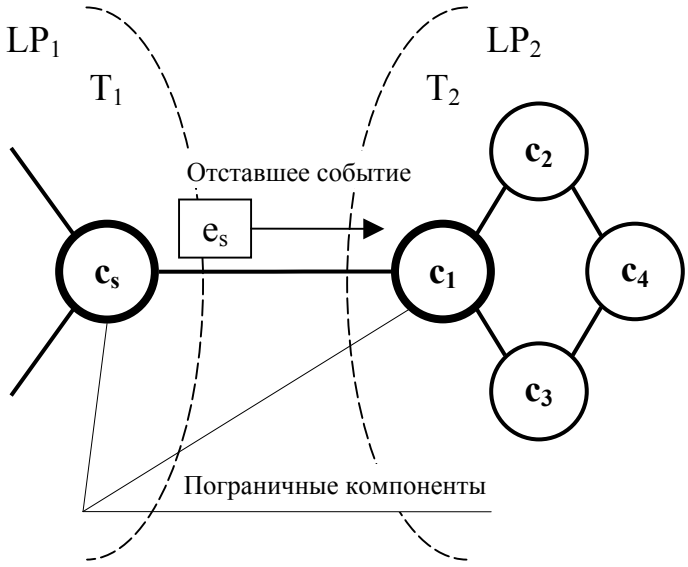

Рис. 1 – Взаимодействие двух логических процессов

Назовем внутрикомпонентным откатом для компонента c откат состояния и отмену событий $e_i=(\tau_i, \sigma_i, c_i, h_i)$ таких, что $\forall i: h_i = c$ *или* $c_i=c$. Пусть $E_{ROLLBACK}(c, \tau) = \{e_i=(\tau_i, \sigma_i, c_i, h_i) \mid h_i = c$ *и* $\tau_i > \tau\}$ – множество событий, отмененных для компонента c, т.е. множество событий, отмененных в процессе внутрикомпонентного отката.

Если не существует такого события $e_i=(\tau_i, \sigma_i, c_i, h_i)$, $e_i \in E_{ROLLBACK}(c_1, \tau_s)$, что $h_i \neq c_1$ *и* $h_i \in C_2$, то компонент c_1 гарантированно не оказывал влияния ни на один компонент множества C_2, кроме самого себя. Как следствие, в алгоритме Time Warp все отмененные события, не принадлежащие множеству $E_{ROLLBACK}(c_1, \tau_s)$, будут в точности повторены. Это означает, что откат этих событий не имеет смысла. В этом заключается ключевая идея предлагаемого алгоритма внутрикомпонентных откатов.

Рассмотрим случай, когда компонент c_1 оказывает влияние на компонент c_2. Это означает, что $\exists\ e_i=(\tau_i,\ \sigma_i,\ c_i,\ h_i),\ e_i \in E_{ROLLBACK}(c_1,\ \tau_s)$: $ci=c_1,\ hi=c_2$. Такие события могут быть причиной изменения состояния компонента c_2 или причиной порождения этим компонентом новых событий. Следовательно, в компоненте c_2 также необходимо выполнить откат.

Если учесть, что $\forall e_i=(\tau_i,\ \sigma_i,\ c_i,\ h_i)\ \ \tau_i \geq \sigma_i$, и учесть связь между событием-причиной и событием-следствием, то можно сказать, что отмененные в c_1 события не могли быть причиной событий, порожденных ранее в компоненте c_2. В общем случае, $\tau_i > \sigma_i$, что дает возможность уменьшить глубину отката для компонента c_2. События $e_i=(\tau_i,\ \sigma_i,\ c_i,\ h_i),\ e_i \in E_{ROLLBACK}(c_1,\ \tau_s)$ могут быть причиной только таких событий $e_j=(\tau_j,\ \sigma_j,\ c_j,\ h_j)$, что $\sigma_j \geq \min_i \tau_i\ \ \forall j$. Следовательно, откат в компоненте c_2 должен быть выполнен до отметки $\min_i \tau_i$. Согласно условию причинно-следственных связей событий, $\min_i \tau_i \geq \tau_s$. В общем случае, выполняется условие неравенства, что является определяющим фактором уменьшения глубины отката в отдельных компонентах.

Аналогичным образом возможно и влияние между другими компонентами. Если факт влияния установлен, то необходимо выполнить те же действия, что описаны выше. Если имеет место обратное влияние, т.е., например, c_1 повлиял на c_2, а c_2 в свою очередь повлиял на c_1, то его рассматривать нет необходимости. Это связано с тем, что откат компонента c_1 уже выполнен, и рассмотрение обратного влияния уже не уменьшит глубину отката.

Чем глубже будет уходить процесс отката от компонента, получившего отставшее событие, тем меньше будет глубина внутрикомпонентного отката (для общего случая). В худшем случае будут отменены события для всех компонентов до момента времени τ_s, что эквивалентно откату в алгоритме Time Warp.

После выполнения отката необходимо повторить моделирование всех отмененных событий $e_i=(\tau_i,\ \sigma_i,\ c_i,\ h_i)$ таких, что $\sigma_i < \tau_s$. Эти события были созданы до того, как произошло событие e_s, Следовательно, событие e_s не может стать причиной отмены в прошлом этих событий.

Следует учесть, что в процессе обработки отставшего события или в результате повторной обработки отмененных событий могут быть порождены новые события, предназначенные для других компонентов. Если откат этих компонентов не был выполнен, то

новые события станут для них отставшими. В этом случае процесс отката следует повторить.

Алгоритм внутрикомпонентных откатов используется в системе распределенного моделирования алгоритмов маршрутизации [3, 4] для повышения скорости обработки дискретных событий.

Заключение. В статье разработан новый алгоритм внутрикомпонентных откатов. Он позволяет ускорить распределенное моделирование за счет уменьшения глубины откатов. Если отставшее событие с временной меткой τ_s должно быть выполнено компонентом, который не оказывал влияния на другие компоненты, то это будет лучший случай. Откат будет выполнен только для одного компонента. Если же компонент был причиной изменения состояния или создания новых событий всех компонентов логического процесса в момент времени τ_s, то результат работы алгоритма будет эквивалентен откату в алгоритме SRADS with Local Rollbacks.

Литература

[1] R. M. Fujimoto, Parallel and Distributed Simulation Systems, Wiley Interscience, 2000.

[2] P. Dickens and P. Reynolds, Jr. SRADS with local rollback. In Distributed Simulation, volume 22, pages 161--164. SCS Simulation Series, Jan. 1990.

[3] Ладыженский Ю.В., Мирецкий А.В. МОДЕЛИРОВАНИЕ АЛГОРИТМОВ ДИНАМИЧЕСКОЙ МАРШРУТИЗАЦИИ // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: Тези доповідей Міжнародної науково-практичної конференції м. Запоріжжя, 13-15 квітня 2006 року / Під заг. ред. Д.М. Пізи. – Запоріжжя: ЗНТУ. 2006. С. 165-167.

[4] Ладыженский Ю.В., Мирецкий А.В. ПРОГРАММНАЯ СИСТЕМА ДЛЯ МОДЕЛИРОВАНИЯ АЛГОРИТМОВ ДИНАМИЧЕСКОЙ МАРШРУТИЗАЦИИ // ІНТЕРНЕТ-ОСВІТА-НАУКА-2006, п’ята міжнародна конференція ІОН-2006, 10-14 жовтня, 2006. Збірник матеріалів конференції. Том 2. – Вінниця: УНІВЕРСУМ-Вінниця. 2006. С. 372-374.

Получено 01.06.07