электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51036. ISSN 2307-0595

Моделирование распространения компьютерных вирусов методом параллельных вычислений

 # 12, декабрь 2015

Рудаков И. В.[1], Емельянов А. А.[1,*]

УДК: 519.711.2

[1]Россия, МГТУ им. Н.Э. Баумана
[*]sasha-stink@mail.ru

Введение

В настоящее время развитие компьютерных сетей позволяет решить большинство вычислительных задач, так как имеет известные преимущества[5]. Одновременно с предоставляемыми преимуществами создание сетей приводит к увеличению опасности для пользователей, связанных с возможностями распространять по сети вредоносные программные обеспечения. Одни из его разновидностей являются компьютерные вирусы, под которыми понимают программы, создающие свои копии (при этом им необязательно совпадать с оригиналом) и внедряющие их в системные области вычислительной машины, файлы, локальных и интернет сетей, а также возможность осуществить другие разрушающие действия. Полученные копии вирусов имеют способность для дальнейшего распространения. Компьютерные вирусы относятся к вредоносным программам [2].

Анализ ранее проведенных исследований [3] в этой области показал, что при моделировании распространения компьютерных вирусов необходимо учитывать следующие факторы:

- скорость распространения вируса;
- особенности структуры сети.

Существующие модели распространения вирусов в компьютерных сетях (SI, SIR [6], AAWP [7], PSIDR [8]) не учитывают топологию сети. В этом заключается их основной недостаток. Хотя модели AAWP [7] и PSIDR [8] разработаны с учетом особенностей компьютерных вирусов.

Математическая модель [4] на основе цепи Маркова для всех узлов учитывает топологию сети, однако при увеличении количества узлов сложность алгоритма построения матрицы переходных вероятностей экспоненциально растет и составляет $O(N^2 2^{2N})$, где N - число компьютеров, 2^{2N} - число элементов построенной матрицы переходов.

Отсюда вытекают два направления решения проблемы:

1) Сведение задачи к полиномиальной;

2) Разработка алгоритма построения модели на основе методов параллельных вычислений.

Наибольший интерес вызывает параллельное программирование, как имеющее достаточно быстрое решение. Кроме того, для операций с матрицами довольно просто применять методы параллельных вычислений, что также позволит повысить скорость производительности.

Метод параллельных вычислений

Следовательно, необходимо разработать метод параллельных вычислений для марковской модели всей сети и исследование полученных результатов работы алгоритмов.

При разработке метода необходимо:

- Проанализировать разработанный ранее программный аппарат на предмет повышения производительности путем разделения основных, наиболее трудоемких операций, таких как построение матрицы перехода.
- Разработать алгоритм, решающий поставленную задачу, с учётом специфики многопоточного программирования.
- Провести сравнительный анализ производительности предыдущей и новой версии, сделать выводы.

На рисунке 1 представлена классификация методов распространения компьютерных вирусов в зависимости от топологии сети с параллельными вычислениями и без них.

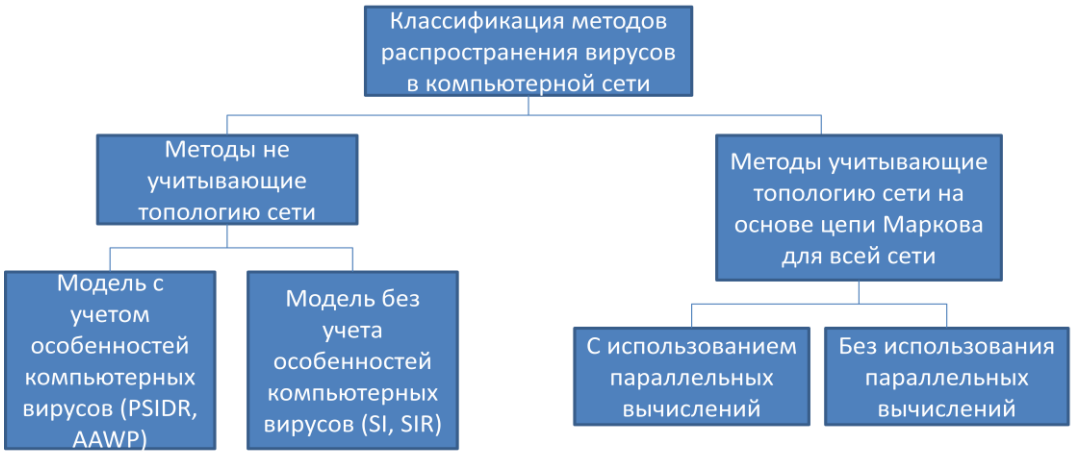

Рис. 1. Классификация используемых методов

Так как модель Маркова учитывает топологию сети и при увеличении количества узлов время вычислений увеличивается, то данная модель была выбрана для проведения распараллеливания вычислений.

Известно, что компьютерные программы — это исполняемые объекты в двоичной (или другой) форме, которые находятся на диске. Процесс — это программа в стадии выполнения, имеющая память, стек данных и собственное адресное пространство. Пока программа лежит на диске - это файл, при запуске - процесс. В течении выполнения процесса

может происходить порождение или ветвление новых процессов для исполнения иных задач, но каждый созданный процесс имеет собственный стек данных, память, адресное пространство и так далее. Итак, возникшие процессы не могут получить доступ к единой информации, если нет реализации межпроцессорного взаимодействия (interprocess communication — IPC) в какой-либо форме. Каждому процессу соответствует свой одиночный поток исполняемых команд и своё адресное пространство.

Потоки подобны процессам, за исключением того, что все они выполняются в пределах одного и того же процесса. Поток - непрерывная часть кода программы, которая может выполняться параллельно с другими частями программы, при этом не имея собственного адресного пространства.

Поток запускается, выполняется в адресном пространстве путем выполнения определенной последовательности и завершается. Созданный поток обладает счетчиком команд, который позволяет следить за тем, где в режиме реального времени происходит его выполнение в текущем положении. Выполнение потока может быть прервано и переведено в состояние ожидания на какой-то промежуток времени (состояние приостановки (sleeping)), в то время как иные потоки продолжат работу. Данная операция имеет называние возврата управления (yielding).

Организованные в одном процессе потоки используют единое пространство данных с основным потоком, поэтому они взаимодействуют или обмениваются друг с другом информацией с меньшими трудностями по сравнению с отдельными процессами. Выполнение потоков обычно происходит параллельно. Как раз распараллеливание и единое использование данных становится фундаментом для выполнения нескольких задач одновременно. В ходе выполнения процесса каждый взятый поток реализует свои конкретные, личные задачи и при получении результатов передает их другим потокам по мере востребования.

Несомненно, что переход от последовательной координации работы к параллельной приводит к возникновению определенных трудностей. Например, для двух или нескольких потоков, которые получают доступ к одной и той же части данных, несмотря на то, в какой зависимости происходит получение доступа, могут появляться несинхронизированные результаты. Неопределенность в получении последовательности доступа называют состоянием состязания (race condition). Но, для решения такой неопределенности в большинстве библиотек, использующих потоки, предусмотрена возможность синхронизации, при которой диспетчер потоков имеет возможность управлять доступом и выполнением.

Ещё одна из сложностей вызвана тем, что нет возможности предоставить всем потокам справедливую и равную часть времени работы. Это происходит из-за того, что некоторые функции блокируют и снимают блокировку только после завершения своего выполнения. Если функцию не разблокировать преднамерено для применения в потоке, то её использование приведет к перераспределению процессорного времени в её пользу. Данные функции называют жадными (greedy).

На рисунке 2 изображена функциональная модель работы метода.

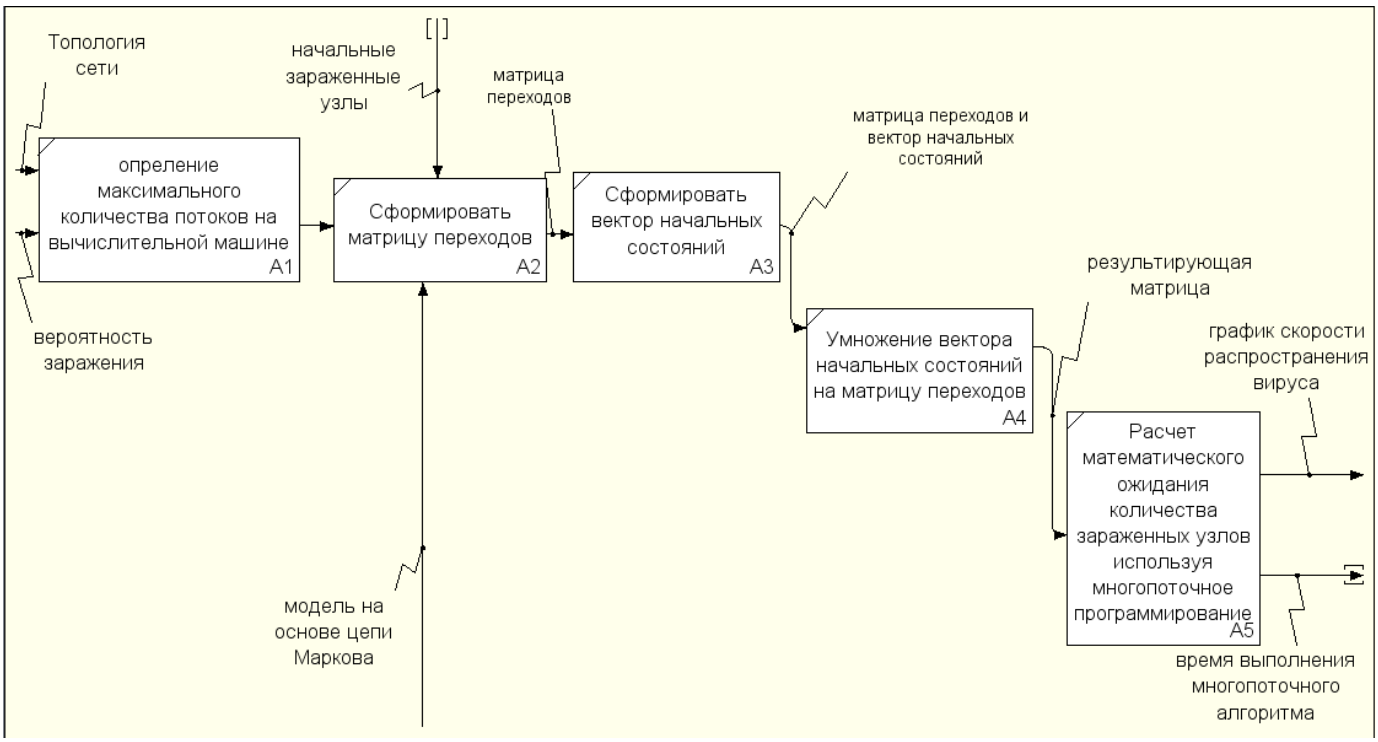

Рис. 2. Функциональная модель работы метода в виде диаграммы IDEF0

Опишем полученную модель. Входными данными метода будут:

- топология сети:
  1) количество компьютеров в сети;
  2) связи между компьютерами в сети (информационные, по средствам которых могут распространяться компьютерные вирусы);
- веса связей (задается вероятность перехода вируса с зараженного компьютера на незараженный за один временной шаг через данные связи);
- выбранная марковская модель.

Выходные данные:

- зависимость количества зараженных узлов от времени (номер шага);
- время выполнения многопоточного алгоритма.

Для упрощения описания алгоритма будем считать, что при построении матрицы перехода каждый узел обрабатывается потоком равно один раз и после завершения работы помечается как обработанный.

1. Определение количества доступных процессоров на используемом компьютере.
2. Если количество используемых в системе процессоров больше одного запуск механизма расчета и формирования потоков. Иначе запуск предыдущего алгоритма построения модели.
3. Цикл от 0 до m, где m, рассчитанное количество потоков.
   a) создать поток;
   b) каждый поток выполняет следующие шаги:

i. выбрать первый необработанный узел;
ii. вычислить вероятности перехода для выбранного элемента матрицы (необработанного узла);
iii. пометить узел как обработанный;
iv. перейти к следующему необработанному узлу.

4. Дождаться завершения работы всех потоков.
5. Сформировать отчет о проделанной работе.

Описанный выше алгоритма построения модели на основе марковских цепей, в идеальном случае, позволит повысить производительность приложения в m раз, где m - число потоков. Данные теоретические сведения требуют практической проверки.

Как было сказано выше, основным недостатком модели на основе цепей Маркова для всей цепи является сложность алгоритма ее построения и, как следствие, недостаточное быстродействие программы на маломощных компьютерах. Для компьютерной сети, состоящей из N компьютеров, существует 2^N разных состояний сети. Для данной модели необходимо строить переходную матрицу, состоящей из 2^{2N} элементов. Так как при расчете вероятности перехода каждого элемента матрицы происходит рекурсивный обход всех компьютеров сети, то алгоритмическая сложность построенной матрицы переходов составит $O(N^2 2^{2N})$.

Данную проблему можно решить путём разделения вычислений на отдельные потоки. Идеальным вариантом является разделение задачи на N потоков.

Тогда построение матрицы переходов для каждого элемента будет выполняться в собственном потоке, что позволит существенно повысить скорость работы алгоритма. В идеальном случае это повышение производительности в m раз, где m - число потоков. На практике такого добиться не получится, так как нужно учитывать особенности межпотокового взаимодействия: взаимоблокировки, проблема совместного доступа к данным, её еще называют гонкой данных, проблема бесконечной отсрочки и так далее.

Однако при этом следует учитывать следующее условие: наиболее эффективное выполнение многопоточных приложений происходит на многопроцессорных машинах (компьютерах). Причем количество потоков не должно превышать количество доступных процессоров.

Для сравнения времени выполнения старой версии алгоритма [3] и разработанного многопоточного алгоритма для модели Маркова для всей сети возьмем одну техническую конфигурацию компьютера: процессор Intel Core i7-4650U, 8 Гб оперативной памяти, ОС Windows 7. Данный процессор может производить вычисления максимум в 4 потока одновременно.

В таблице 1 приведены полученные данные исследования времени работы алгоритма в зависимости от количества узлов в сети. Сравнение идет между однопоточной и многопоточной версиями разработанных алгоритмов, а также идеальной версией теоретического подсчета при многопоточном алгоритме.

Таблица 1. Результаты проведенного исследования по времени работы алгоритмов

Количество узлов	Однопоточный алгоритм, сек.	Многопоточный алгоритм, сек.	Идеальная (теоретическая) работа многопоточного алгоритма, сек.
5	0,5	0,5	0,125
10	4,288	1,429	1,072
12	118,931	39,644	29,733
15	360,774	120,258	90,194

Как видно из таблицы 1 разность во времени моделирования возрастает с увеличением количества узлов сети. Применение алгоритмов параллельных вычислений позволяет заметно повысить производительность системы при количестве узлов $N \geq 10$. Для меньшего количества узлов возможно незначительное ухудшение результатов, так как часть времени уходит на формирование потоков и ожидание ими очереди. Поэтому в настройки приложения добавлена возможность указывать минимальное количество узлов, при котором будет запускаться многопоточный алгоритм. По умолчанию данное значение равно 10.

Заключение

Разработанный метод параллельных вычислений для модели цепи Маркова позволяет значительно сократить время выполнения моделирования, в отличии от однопоточного [3]. Следовательно, многопоточное программирование нужно применять при ёмких математических вычислениях.

Еще одним преимуществом многопоточного метода является его хорошая адаптивность к многопроцессорным системам.

Список литературы

[1]. Бабанин Д.В. Модели распространения компьютерных вирусов на основе цепей Маркова // Математическое и программное обеспечение вычислительных систем: Меж вуз. сб. науч. тр. / Под ред. А.Н. Пылькина. М.: Горячая линия – Телеком. 2009. 156 с. С. 89-93.

[2]. ГОСТ Р 51188-98. Защита информации. Испытания программных средств на наличие компьютерных вирусов. Типовое руководство. Принят и введен в действие 14.07.1998. № 295. М.: ИПК Издательство стандартов. 1998. 9 с.

[3]. Емельянов А. А. Анализ распространения вирусов в компьютерных сетях на основе цепи Маркова // Молодежный научно-технический вестник. Электронный журнал МГТУ им. Н.Э. Баумана. 2015. №8. Режим доступа: http://sntbul.bmstu.ru/doc/799648.html (дата обращения 17.11.2015).

[4]. Рудаков И.В. Представление формальной макромодели функционального блока сложной дискретной структуры в виде логической сети // Инженерный вестник. Электронный научно-технический журнал МГТУ им. Н.Э. Баумана. 2014. № 10. Режим доступа: http://engbul.bmstu.ru/doc/730829.html (дата обращения 16.11.2015).

[5]. Таненбаум Э. Компьютерные сети. 4-е издание. СПб.: Питер. 2010. 992 с.

[6]. Kephart J.O., White S.R. Directed-Graph Epidemiological Models of Computer Viruses // IEEE Symposium on Security and Privacy. 1991. P. 343-361. DOI: 10.1109/RISP.1991.130801.

[7]. Chen Z., Gao L., Kwait K. Modeling the spread of active worms. // CONFERENCE. IEEE INFOCOM. 2003. Vol.3. P.1890-1900. Режим доступа: http://infocom2003.ieee-infocom.org/papers/46_03.PDF (дата обращения 20.03.2015).

[8]. Williamson M.M., Leveille J. An epidemiological model of virus spread and cleanup. Copyright Hewlett-Packard Company. 2003. Режим доступа: http://www.hpl.hp.com/techreports/2003/HPL-2003-39.pdf (дата обращения 20.03.2015).