УДК 519.673
DOI: 10.15827/0236-235X.132.557-563
Дата подачи статьи: 29.09.20
2020. Т. 33. № 4. С. 557–563

Исследование алгоритма многократной маркировки перколяционных кластеров при частичной загрузке вычислительных узлов на суперкомпьютерных системах

С.Ю. Лапшина [1], *старший научный сотрудник, lapshina@jscc.ru*
А.Н. Сотников [1], *д.ф.-м.н., профессор, главный научный сотрудник, asotnikov@jscc.ru*
В.Е. Логинова [1], *ведущий инженер-программист, vl@jscc.ru*

[1] *Межведомственный суперкомпьютерный центр РАН (МСЦ РАН) – филиал ФНЦ НИИСИ РАН, г. Москва, 119991, Россия*

В статье рассматривается поведение алгоритма многократной маркировки перколяционных кластеров в ходе проведения имитационных экспериментов задачи мультиагентного моделирования процессов распространения массовых эпидемий при частичной загрузке запрашиваемых вычислительных узлов современных суперкомпьютерных систем, установленных в Межведомственном суперкомпьютерном центре Российской академии наук (МСЦ РАН).

Алгоритм многократной маркировки перколяционных кластеров – универсальное средство, которое может быть использовано в любой области в качестве инструмента дифференцирования кластеров решетки большого размера. На вход он получает данные в формате, не зависящем от приложения. Так, в МСЦ РАН этот инструмент был использован для изучения задачи распространения эпидемий. Возможно применение данного алгоритма для изучения поведения нефтяных пластов, процессов протекания воды через пористые материалы, распространения лесных пожаров и многого другого.

В ходе имитационных экспериментов применялся усовершенствованный на многопроцессорной системе вариант алгоритма многократной маркировки перколяционных кластеров Хошена–Копельмана, связанный с механизмом линковки меток.

В статье проводится сравнительный анализ времени выполнения алгоритма многократной маркировки перколяционных кластеров Хошена–Копельмана при частичной и полной загрузке вычислительных узлов и при различных значениях входных параметров на четырех основных высокопроизводительных вычислительных системах, установленных в МСЦ РАН – суперкомпьютерах МВС-10П МП2 KNL, МВС-10П ОП, МВС 10П Торнадо, МВС-100К.

Ключевые слова: *мультиагентное моделирование, перколяционный кластер, механизм линковки меток, высокопроизводительные вычислительные системы, вычислительный узел, процессорные ядра.*

Алгоритм *многократной маркировки перколяционных кластеров* (ММПК) Хошена–Копельмана – один из базовых алгоритмов теории перколяции, который позволяет выделять связные кластеры (подграфы) некоторой случайной решетки (графа) за один проход.

Решетка формируется в ходе имитационного эксперимента и, как правило, имеет большой размер. Возникает необходимость ее хранения и обработки в параллельном режиме, зачастую на многопроцессорных системах, поэтому разработка специализированных методов распараллеливания и изучения их эффективности становится достаточно актуальной.

Описание алгоритма

Рассмотрим собственно алгоритм ММПК. В начале его работы всем занятым узлам решетки приписываются различные кластерные метки. Принадлежность узла к тому или иному кластеру является глобальным свойством и может быть определена только после просмотра всей решетки.

Узлы решетки нумеруются. Номер узла – начальное значение его метки. При последовательном обходе решетки слева направо, затем сверху вниз для каждого узла рассматриваются связанные с ним соседние узлы. Каждой группе (текущий узел, соседние с ним узлы) ставится ее минимальная метка. На каждом шаге все производимые замены меток должны отражаться на всех узлах решетки, то есть, если на некотором шаге метка одного узла была заменена, надо заменить и все остальные метки узлов, равные данной [1–4].

В результате работы алгоритма ММПК все узлы решетки будут разделены по их принад-

лежности к тому или иному кластеру (принадлежность определяется меткой – все узлы с одинаковыми метками принадлежат одному и тому же графу). На рисунке 1 приведен пример работы алгоритма Хошена–Копельмана.

При наличии у компьютера большого количества процессоров выгоднее использовать параллельный алгоритм ММПК [4–6]. Каждому процессору назначается группа узлов решетки. Параллельный вариант алгоритма совпадает с обычным алгоритмом ММПК за одним исключением: вместо прохода по всей решетке каждый процессор выполняет действия алгоритма ММПК только на назначенной ему группе узлов с последующим обменом информацей между процессорами. Алгоритм завершается тогда, когда на очередном шаге после обмена информацией метки всех групп узлов перестают изменяться.

Важным моментом в работе алгоритма является создание механизма линковки меток. Во время работы алгоритма ММПК при последовательном прохождении узлов на каждом из них при различии меток данного узла и его соседей приходится заменять метки этих узлов на минимальную среди них. Все замененные таким образом метки должны быть также заменены среди всех остальных меток решетки: если, например, данный узел имеет метку F, а его сосед метку Y, то ему и его соседу присваивается метка $\min(F, Y)$, при этом надо заменить и метки всех узлов решетки, имеющие значение F или Y, на $\min(F, Y)$. Таким образом, время работы алгоритма зависит от размера решетки N как $O(N^2)$. Механизм линковки меток позволяет добиться скорости $O(N)$ [4, 6].

При распараллеливании алгоритма ММПК важным является правильный подбор количества процессорных ядер, на которых будет производиться обработка исходной решетки.

Решетка при работе алгоритма загружается в оперативную память узла, и, принимая во внимание ее большой размер, логично проводить распараллеливание процесса ее обработки на большое количество частей.

Но, с другой стороны, в ходе работы алгоритма нужно проводить обмен данными между пограничными ячейками частей исходной решетки. Время обмена данными может превысить отведенный лимит времени на обработку задания, если таких частей будет слишком много.

Нахождение баланса между увеличением количества запрашиваемых вычислительных ядер и задержками, связанными с обменом данными между пограничными ячейками, является важной задачей при запуске алгоритма на суперкомпьютере [4].

Также представляет интерес поведение алгоритма при использовании не всех процессоров вычислительного узла целиком, а лишь его части, распределяя кусок решетки в оперативной памяти всего узла.

Ввиду высокой стоимости суперкомпьютерного времени к качеству распределения ресурсов предъявляются повышенные требования по минимизации времени простоя вычислительных ресурсов. Подобные исследования проводятся для широкого круга задач [7–9].

1	0	0	0	1	1	1	1	1	1	0	1	1	1	0
0	1	0	0	1	0	1	1	1	0	0	0	1	0	1
0	1	0	0	1	0	1	1	1	0	0	0	1	0	1
1	1	0	0	1	0	1	1	1	1	1	1	1	0	1
1	1	0	0	1	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	0	0	0	1	1	1	1	0	0
0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
0	0	1	1	1	0	0	0	0	0	0	0	1	0	0
0	0	1	1	1	1	0	0	0	0	0	0	1	1	0
0	1	1	1	0	0	1	1	1	0	0	0	0	0	1
0	1	1	1	1	0	1	1	1	0	0	0	1	0	1
1	1	1	1	1	0	1	1	1	1	1	1	1	0	1
1	1	1	1	1	0	1	1	1	1	1	1	1	0	1
1	0	1	1	1	0	0	0	0	1	1	1	1	0	0

1	0	0	2	2	2	2	2	2	2	0	2	2	2	0
0	4	0	0	2	0	2	2	2	0	0	0	2	0	5
0	4	0	0	2	0	2	2	2	0	0	0	2	0	5
4	4	0	0	2	0	2	2	2	2	2	2	2	0	5
4	4	0	0	2	0	2	2	2	2	2	2	2	0	5
4	0	0	0	2	0	0	0	0	2	2	2	2	0	0
0	0	0	0	2	0	0	0	0	0	0	0	2	0	0
0	0	0	0	2	0	0	0	0	0	0	0	2	0	0
0	0	2	2	2	0	0	0	0	0	0	0	2	0	0
0	0	2	2	2	2	0	0	0	0	0	0	2	2	0
0	2	2	2	0	0	6	6	6	0	0	0	0	0	7
0	2	2	2	2	0	6	6	6	0	0	0	6	0	7
2	2	2	2	2	0	6	6	6	6	6	6	6	0	7
2	2	2	2	2	0	6	6	6	6	6	6	6	0	7
2	0	2	2	2	0	0	0	0	6	6	6	6	0	0

Рис. 1. Исходная и конечная решетки при работе алгоритма Хошена–Копельмана

Fig. 1. Original and final lattices in the operation of the Hoshen–Kopelman algorithm

Суперкомпьютерные системы для проведения исследования

Программа маркировки кластеров Load, реализующая параллельный алгоритм многократной маркировки перколяционных кластеров, запускалась на семи основных суперкомпьютерных системах, установленных в МСЦ РАН [10]: МВС-10П ОП (разделы Cascadelake, Haswell, Broadwell и Skylake), МВС-10П МП2, МВС-10П Торнадо, МВС-100К.

Программа запускалась с входным параметром вероятности p от 0,01 до 1 с шагом в 0,01 при постоянных значениях модельного времени t = 30 дней на 48–172 (в зависимости от системы) процессорных ядрах при 100- и 50-процентной загрузке вычислительных узлов (по числу процессоров узла).

Исследование поведения алгоритма при 100-процентной загруженности узлов на всех указанных системах, за исключением суперкомпьютера МВС-10П ОП Cascadelake (запущен в эксплуатацию в 2020 году), осуществлено в [4].

Рассмотрим вычислительные характеристики суперкомпьютеров (их разделов) подробнее.

МВС-10П ОП предоставляется пользователям Центра в режиме коллективного доступа к четырем разделам – Cascadelake, Haswell, Broadwell и Skylake:

– Cascadelake; 51 вычислительный модуль на базе процессоров Intel Xeon Platinum 8268 (Cascade Lake), 192 Гб оперативной памяти на модуль, пиковая производительность модуля – 4,454 tflops, 2 448 ядер в разделе;

– Haswell; 42 вычислительных модуля на базе процессоров Intel Xeon E5-2697 v3, 128 Гб оперативной памяти на модуль, пиковая производительность модуля – 1,1648 tflops, 1 176 ядер в разделе;

– Broadwell; 136 вычислительных модулей на базе процессоров Intel Xeon E5-2697 v4, 128 Гб оперативной памяти на модуль, пиковая производительность модуля – 1,3312 tflops, 4 352 ядра в разделе;

– Skylake; 58 вычислительных модулей на базе процессоров Intel Xeon Gold 6154, 192 Гб оперативной памяти на модуль, пиковая производительность модуля – 3,456 tflops, 2 088 ядер в разделе.

Общим для установок на МВС-10П ОП является использование в качестве коммуникационной среды низколатентной сети Intel Omni-Path.

МВС-10П МП2 KNL – суперкомпьютер из 38 вычислительных модулей на базе процессоров Intel Xeon Phi 7290, 96 ГБ оперативной памяти на модуль, пиковая производительность модуля – 3,456 tflops, 2 736 ядер в системе.

МВС-10П Торнадо – суперкомпьютер из 207 вычислительных модулей, каждый модуль имеет в своем составе 2 процессора Xeon E5-2690, 64 ГБ оперативной памяти, два сопроцессора Intel Xeon Phi 7110X, пиковая производительность модуля – 371,2 gflops, 3 312 ядер в системе.

МВС-100К – суперкомпьютер из 110 вычислительных модулей на базе процессоров Intel Xeon E5450, 8 ГБ оперативной памяти на модуль, пиковая производительность модуля – 96 gflops, 880 ядер в системе.

Анализ поведения алгоритма при частичной и полной загруженности вычислительных узлов

График зависимости времени работы программы Load от количества запрашиваемых процессорных ядер на различных разделах МВС-10П ОП при 100- и 50-процентной загруженности вычислительных узлов показан на рисунке 2. На Cascadelake исследование проводилось при ограниченном числе ядер (48–96), данное ограничение связано с доступностью ядер для пользователя по мере ввода нового раздела суперкомпьютера в эксплуатацию.

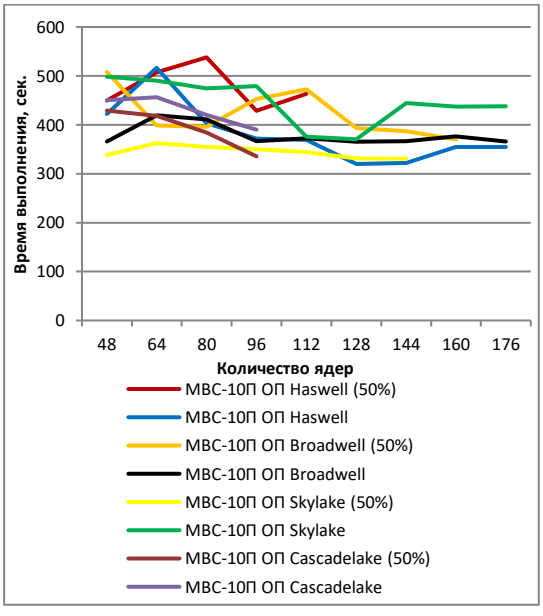


Рис. 2. Расчет на разделах МВС-10П ОП

Fig. 2. Calculation on sections MVS-10P OP

При 100-процентной загруженности узла на МВС-10П ОП среднее время расчета составило:

– раздел Cascadelake – 429 сек.;
– раздел Haswell – 382 сек.;
– раздел Broadwell – 379 сек.;
– раздел Skylake – 445 сек.

Минимальное время запуска:

– раздел Cascadelake – 390 сек. на 96 ядрах;
– раздел Haswell – 320 сек. на 128 ядрах;
– раздел Broadwell – 366 сек. на 208 ядрах;
– раздел Skylake – 370 сек. на 128 ядрах.

При 50-процентной загруженности узла на МВС-10П ОП среднее время расчета составило:

– раздел Cascadelake – 392 сек.;
– раздел Haswell – 477 сек.;
– раздел Broadwell – 422 сек.;
– раздел Skylake – 345 сек.

Минимальное время запуска:

– раздел Cascadelake – 336 сек. на 96 ядрах;
– раздел Haswell – 428 сек. на 96 ядрах;
– раздел Broadwell – 370 сек. на 160 ядрах;
– раздел Skylake – 331 сек. на 128 ядрах.

Как видно из представленного графика, экономии времени выполнения программы на разделах на МВС-10П ОП при использовании 50-процентной загрузки узлов нет.

На рисунке 3а показан график зависимости времени работы программы Load от количества запрашиваемых процессорных ядер на МВС-10П МП2 KNL при 100-процентной и 50-процентной загруженности вычислительных узлов.

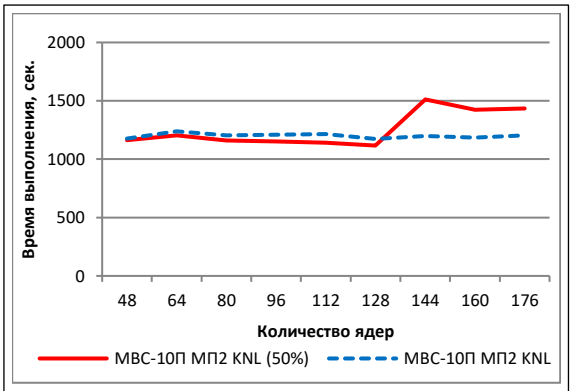
Среднее время расчета составило 1 200 сек. при 100-процентной загрузке узлов и 1 256 сек. при 50-процентной, минимальное время запуска – 1 173 сек. на 128 ядрах при 100-процентной загрузке узлов, 1 117 сек. на 128 ядрах при 50-процентной.

Представленный график показывает, что экономии времени выполнения программы на МВС-10П МП2 KNL, как и в случае МВС-10П ОП, при использовании 50-процентной загрузки узлов нет.

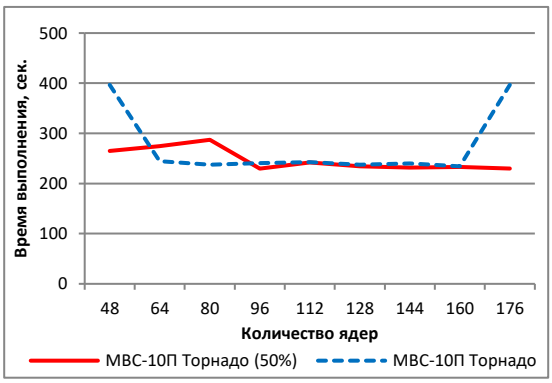
На рисунке 3б показан график зависимости времени работы программы Load от количества запрашиваемых процессорных ядер на МВС-10П Торнадо при 100-процентной и 50-процентной загруженности вычислительных узлов.

Среднее время расчета составило 275 сек. при 100-процентной загрузке узлов и 247 сек. при 50-процентной, минимальное время запуска – 234 сек. на 160 ядрах при 100-процентной загрузке узлов, 230 сек. на 96 ядрах при 50-процентной.

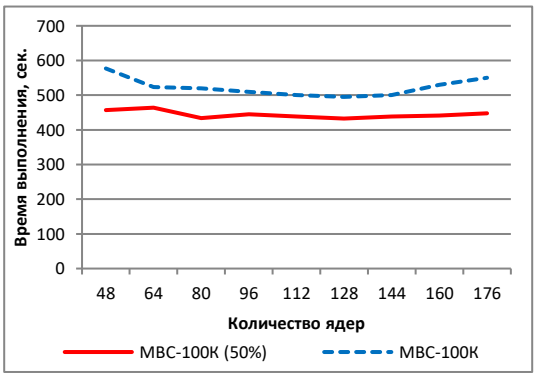
На МВС-10П Торнадо экономии времени выполнения программы при использовании 50-процентной загрузки узлов нет.



а)



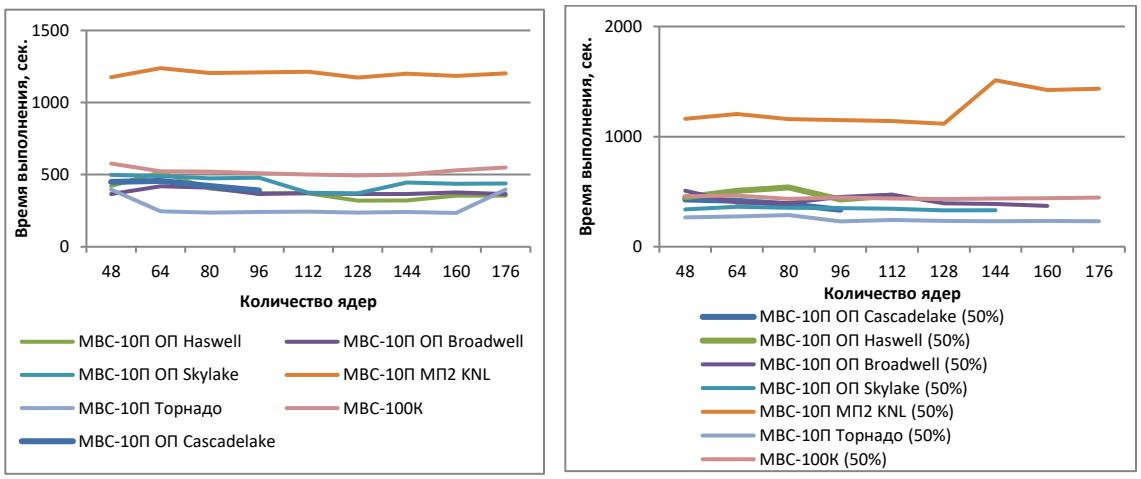
б)



в)

Рис. 3. Результаты расчета:
а) на МВС-10П МП2 KNL,
б) на МВС-10П Торнадо, в) на МВС-100К

Fig. 3. Calculation results:
a) on MVS-10P MP2 KNL,
б) on MVS-10P Tornado, в) on MVS-100K



а) *б)*

Рис. 4. Сводный график зависимости времени расчета от количества запрашиваемых процессорных ядер: а) при 100-процентной загрузке вычислительных узлов, б) при 50-процентной загрузке вычислительных узлов

Fig. 4. A summary graph of the dependence of the calculation time on the number of requested processor cores: a) at 100 % load of compute nodes, б) at 50 % load of computing nodes

На рисунке 3в показан график зависимости времени работы программы Load от количества запрашиваемых процессорных ядер на МВС-100К при 100-процентной и 50-процентной загруженности вычислительных узлов.

Среднее время расчета составило 523 сек. при 100-процентной загрузке узлов и 444 сек. при 50-процентной, минимальное время запуска – 495 сек. на 128 ядрах при 100-процентной загрузке узлов, 432 сек. на 128 ядрах при 50-процентной.

Как видно из представленного графика, на МВС-100К наблюдается незначительная экономия времени выполнения программы при использовании 50-процентной загрузки узлов.

На рисунке 4а представлен сводный график зависимости времени работы программы Load от количества запрашиваемых процессорных ядер на основных системах МСЦ РАН при 100-процентной загруженности вычислительных узлов. Минимальное время расчета показывает МВС-10П Торнадо. Для большинства суперкомпьютеров минимальное время счета достигается при использовании 128–208 ядер.

На рисунке 4б приведен сводный график зависимости времени работы программы Load от количества запрашиваемых процессорных ядер на основных системах МСЦ РАН при 50-процентной загруженности вычислительных узлов. Минимальное время расчета показывает МВС-10П Торнадо. Для большинства суперкомпьютеров минимальное время счета достигается при использовании 128 ядер, как и в случае 100-процентного использования вычислительного узла.

Заключение

В статье приведен анализ времени выполнения алгоритма многократной маркировки перколяционных кластеров Хошена–Копельмана при частичной (50 %) и полной загрузке вычислительных узлов. Существенной экономии времени выполнения при частичной загрузке на большинстве систем не обнаружено.

Расчеты проводились на высокопроизводительных вычислительных системах МВС-10П МП2 KNL, МВС-10П ОП, МВС 10П Торнадо, МВС-100К в МСЦ РАН.

Работа выполнена в МСЦ РАН в рамках государственного задания № 0065-2019-0014 и проекта РФФИ № 19-07-00861.

Литература

1. Hoshen J., Kopelman R. Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm. Phys. Rev. B, 1976, vol. 14, no. 8, pp. 3438–3445. DOI: 10.1103/PhysRevB.14.3438.

2. Тарасевич Ю.Ю. Перколяция: теория, приложения, алгоритмы. М.: УРСС, 2002. 112 с.

3. Казаков С.А., Шебеко Ю.А. Введение в практику имитационного моделирования и анализа поведения сложных процессов и систем. М.: МИЭТ, 2006. 147 с.

4. Сотников А.Н., Лапшина С.Ю., Логинова В.Е., Юдинцев К.Ю. Исследование оптимального количества процессорных ядер для алгоритма многократной маркировки перколяционных кластеров на суперкомпьютерных вычислительных системах // Программные продукты и системы. 2019. № 4. С. 573–580. DOI: 10.15827/0236-235X.128.573-580.

5. Утакаева И.Х. Имитационное моделирование распространения эпидемий на основе агентного подхода // Научный журнал КубГАУ. 2016. № 121. DOI: 10.21515/1990-4665-121-085. URL: http://ej.kubagro.ru/2016/07/pdf/85.pdf (дата обращения: 10.09.2020).

6. Lapshina S.Yu. High-performance computations in multi-agent simulation problems of percolation cluster's behavior. Lobachevskii Journal of Mathematics, 2019, vol. 40, no. 3, pp. 341–348. DOI: 10.1134/S1995080219030144.

7. Savin G.I., Benderskiy L.A., Lyubimov D.A., Rybakov A.A. RANS/ILES method optimization for effective calculations on supercomuter. Lobachevskii Journal of Mathematics, 2019, vol. 40, no. 5, pp. 566–573. DOI: 10.1134/S1995080219050172.

8. Рыбаков А.А. Распределение вычислительной нагрузки между узлами гетерогенного вычислительного кластера // Программные продукты, системы и алгоритмы. 2018. Т. 6. № 1. URL: http://swsys-web.ru/distribution-of-computational-load-between-cluster-nodes.html (дата обращения: 10.09.2020). DOI: 10.15827/2311-6749.26.300.

9. Рыбаков А.А. Двухуровневое распараллеливание для оптимизации вычислений на суперкомпьютере при расчете задач газовой динамики // Вычислительный эксперимент в аэроакустике: сб. тезисов VI Всерос. конф. 2016. С. 214–217.

10. Savin G.I., Shabanov B.M., Telegin P.N., Baranov A.V. Joint Supercomputer Center of the Russian Academy of Sciences: Present and Future. Lobachevskii Journal of Mathematics, vol. 40, no. 11, pp. 1853–1862. DOI: 10.1134/S1995080219110271.

Software & Systems
DOI: 10.15827/0236-235X.132.557-563

Received 29.09.20
2020, vol. 33, no. 4, pp. 557–563

Algorithm analysis for multiple marking of percolation clusters with a partial load of computing nodes on supercomputer systems

S.Yu. Lapshina [1], *Senior Researcher, lapshina@jscc.ru*
A.N. Sotnikov [1], *Dr.Sc. (Physics and Mathematics), Professor, Chief Researcher, asotnikov@jscc.ru*
V.E. Loginova [1], *Leading Engineer-Programmer, vl@jscc.ru*

[1] *Joint Supercomputer Center of RAS – Branch of Federal State Institution "Scientific Research Institute for System Analysis of the Russian Academy of Sciences" (SRISA RAS), Moscow, 119991, Russian Federation*

Abstract. The paper considers the behavior of the Parallel Cluster Multiple Marking Technique in the course of simulation experiments on the problem of multi-agent modeling with a partial load of the requested computing nodes of modern supercomputer systems installed in the JSCC RAS.

The Cluster Multiple Marking Technique is a universal tool that can be used in any field as a tool for differentiating large lattice clusters. It receives data as input in an application-independent format.

So, at the JSCC RAS, this tool was used to study the problem of spreading epidemics. It is possible to use this technique to study the behavior of oil reservoirs, the processes of water flow through porous materials, study the spread of forest fires, and much more.

In the course of simulation experiments, the authors applied a version of the algorithm for multiple making of Hoshen – Kopelman percolation clusters, which was improved on a multiprocessor system, and associated with the linking mechanism of labels.

The paper provides a comparative analysis of algorithm execution time of Hoshen – Kopelman multiple labeling of percolation clusters and partial and a full load of computational nodes and various values of input parameters on four main high-performance computing systems installed in the JSCC RAS: MVS-10P MP2 KNL, MVS-10P OP, MVS 10P Tornado, MVS-100K.

Keywords: multi-agent simulation, percolation's cluster, parallel cluster multiple making technique, high-performance computing systems, processor cores.

Acknowledgements. *The reported study was funded by JSCC RAS within the framework of state assignment no. 0065-2019-0014 and RFBR project no. 19-07-00861.*

References

1. Hoshen J., Kopelman R. Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm. *Phys. Rev. B*, 1976, vol. 14, no. 8, pp. 3438–3445. DOI: 10.1103/PhysRevB.14.3438.
2. Tarasevich Yu.Yu. *Percolation: Theory, Applications, Algorithms*. Moscow, 2002, 64 p. (in Russ.).
3. Kazakov S.A., Shebeko Ju.A. *Introduction to the Practice of Simulation and Analysis of the Behavior of Complex Processes and Systems*. Moscow, 2006, 147 p. (in Russ.).
4. Sotnikov A.N., Lapshina S.Yu., Loginova V.E., Yudintsev K.Yu. Investigation of the optimal number of processor cores for parallel cluster multiple labeling on supercomputers. *Software & Systems*, 2019, no. 4, pp. 573–580 (in Russ.). DOI: 10.15827/0236-235X.128.573-580.
5. Utakaeva I.Kh. Simulation modeling of distribution of epidemics on the basis of agent approach. *Polythematic Online Scientific Journal of Kuban State Agrarian University*, 2016, no. 121. Available at: http://ej.kubagro.ru/2016/07/pdf/85.pdf (accessed September 10, 2020) (in Russ.). DOI: 10.21515/1990-4665-121-085.
6. Lapshina S.Yu. High-performance computations in multi-agent simulation problems of percolation cluster's behavior. *Lobachevskii Journal of Mathematics*, 2019, vol. 40, iss. 3, pp. 341–348. DOI: 10.1134/S1995080219030144.
7. Savin G.I., Benderskiy L.A., Lyubimov D.A., Rybakov A.A. RANS/ILES method optimization for effective calculations on supercomuter. *Lobachevskii Journal of Mathematics*, 2019, vol. 40, no. 5, pp. 566–573. DOI: 10.1134/S1995080219050172.
8. Rybakov A.A. Distribution of computational load between nodes of a heterogeneous computational cluster. *Software Journal: Theory and Applications*, 2018, vol. 6, no. 1. Available at: http://swsys-web.ru/distribution-of-computational-load-between-cluster-nodes.html (accessed September 10, 2020) (in Russ.). DOI: 10.15827/2311-6749.26.300.
9. Rybakov A.A. Two-level parallelization to optimize supercomputer computations when calculating gas dynamics problems. *Proc. 6th Intern. Conf. "Computational Experiment in Aeroacoustics"*, 2016, pp. 214–217 (in Russ.).
10. Savin G.I., Shabanov B.M., Telegin P.N., Baranov A.V. Joint Supercomputer Center of the Russian Academy of Sciences: Present and Future. *Lobachevskii Journal of Mathematics*, vol. 40, no. 11, pp. 1853–1862 (in Russ.). DOI: 10.1134/S1995080219110271.

Для цитирования

Лапшина С.Ю., Сотников А.Н., Логинова В.Е. Исследование алгоритма многократной маркировки перколяционных кластеров при частичной загрузке вычислительных узлов на суперкомпьютерных системах // Программные продукты и системы. 2020. Т. 33. № 4. С. 557–563. DOI: 10.15827/0236-235X.132.557-563.

For citation

Lapshina S.Yu., Sotnikov A.N., Loginova V.E. Algorithm analysis for multiple marking of percolation clusters with a partial load of computing nodes on supercomputer systems. *Software & Systems*, 2020, vol. 33, no. 4, pp. 557–563 (in Russ.). DOI: 10.15827/0236-235X.132.557-563.