УДК 519.673
DOI: 10.15827/0236-235X.136.589-596
Дата подачи статьи: 24.06.21
2021. Т. 34. № 4. С. 589–596

Сравнительный анализ работы алгоритма многократной маркировки перколяционных кластеров на различных разделах суперкомпьютера МВС-10П ОП

С.Ю. Лапшина [1], *старший научный сотрудник, lapshina@jscc.ru*

[1] *Межведомственный суперкомпьютерный центр РАН – филиал Федерального научного центра Научно-исследовательского института системных исследований РАН, г. Москва, 119334, Россия*

В статье проведен сравнительный анализ работы алгоритма многократной маркировки перколяционных кластеров на пяти различных разделах суперкомпьютера МВС-10П ОП (с учетом добавления в 2021 г. нового раздела и модернизации существующих), установленного в Межведомственном суперкомпьютерном центре Российской академии наук.

Алгоритм многократной маркировки перколяционных кластеров используется в Центре для изучения процессов распространения эпидемий. Вместе с тем это универсальное средство, которое может найти применение в любой области в качестве инструмента дифференцирования кластеров решетки большого размера, получающее на вход данные в формате, не зависящем от приложения. Известны разработки с использованием данного алгоритма для изучения процессов протекания воды через пористые материалы, поведения нефтяных пластов, распространения лесных пожаров.

При суперкомпьютерном имитационном эксперименте применялся усовершенствованный для применения на многопроцессорной системе вариант алгоритма многократной маркировки перколяционных кластеров Хошена–Копельмана, связанный с механизмом линковки меток.

В статье сравнивается время выполнения алгоритма многократной маркировки перколяционных кластеров Хошена–Копельмана при полной загрузке вычислительных узлов и различных значениях входных параметров на пяти разделах – Broadwell, Cascadelake, Skylake, Optan, KNL суперкомпьютера МВС-10П ОП. Установлено оптимальное количество процессорных ядер для вычислений.

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

Исторически процессы перколяции впервые привлекли к себе внимание исследователей сложного поведения структур формирования гелей при полимеризации. В русскоязычных источниках можно столкнуться с терминами «теория перколяции», «теория протекания» и даже «теория просачивания». Появление подобных терминов обусловлено тем, что первые исследования в этом направлении были посвящены процессам просачивания (протекания) жидкостей или газов через пористую среду. До сих пор подобные работы занимают существенный объем в соответствующих приложениях теории. Понятие перколяционного процесса нередко используется физиками для противопоставления процессам диффузии: если в случае диффузии мы имеем дело со случайным блужданием частицы в среде регулярной, то в случае перколяции речь идет о регулярном движении (течение жидкости или протекание электрического тока) в среде случайной. Теория и практика моделирования перколяционных процессов имеют точки соприкосновения с рядом новых и перспективных направлений, таких как исследования процессов самоорганизации и образования фрактальных структур, а также актуальные исследования поведения широкого класса процессов и явлений, которые принято называть критическими. Результативность подобных работ определяется, как правило, проведением чрезвычайно большого количества имитационных экспериментов, разработкой специфических и эффективных алгоритмов, компьютерным моделированием поведения объектов с привлечением вычислительных возможностей современных суперкомпьютеров.

Базовый алгоритм теории перколяции

Алгоритм *многократной маркировки кластеров Хошена–Копельмана* (ММК-алго-

ритм) [1–4], предложенный в 1976 г., позволяет выделять связные подграфы (кластеры) некоторого случайного графа. Важной особенностью алгоритма является его однопроходность. За один проход алгоритм позволяет выяснить, к какому кластеру относится тот или иной узел решетки, и распределяет кластеры по размерам.

Идея алгоритма заключается в том, что всем занятым узлам решетки приписываются различные кластерные метки, и основана она на соображении, что принадлежность узла к тому или иному кластеру является глобальным свойством и может быть определена только после просмотра всей решетки.

Рассмотрим работу алгоритма на решетке, узлы которой заполнены с вероятностью p, где $0 < p < 1$. Данная решетка формируется при помощи генератора случайных чисел на интервале [0, 1]. Если сгенерированное число меньше или равно p, то очередному элементу массива присваивается значение 0, иначе 1.

Узлы решетки нумеруются. Номер узла – начальное значение его метки. При последовательном обходе решетки для каждого узла рассматриваются связанные с ним соседние узлы. Каждой группе (текущий узел, соседние с ним узлы) ставится минимальная метка этой группы. 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 все остальные метки узлов, равные данной. Псевдокод алгоритма Хошена–Копельмана будет следующим:

```
макс_метка = 0;
for x from 0 to n_колонок {
 for y from 0 to n_рядов {
 if A[x, y] != 0 then
 слева = A[x-1, y];
 сверху = A[x, y-1];
 if (слева == 0) and (сверху == 0) then
 макс_метка = макс_метка + 1;
 A_с_метками[x, y] = макс_метка;
 else {
 if (слева != 0) {
 if (справа != 0)
 объединить (слева, сверху);
 A_с_метками [x, y] = найти(сверху);
 } else
 A_с_метками [x, y] = найти(справа);
 }
 }
 },
```

где $A[x, y]$ – исходный массив; A_с_метками $[x, y]$ – конечный массив; объединить(x, y) – команда присвоения узлу y метки узла x; найти(x) – команда нахождения ближайшей ячейки того же кластера (ячейки с той же меткой), что и x.

В результате работы алгоритма ММК все узлы решетки будут разделены по их принадлежности к тому или иному кластеру (принадлежность определяется меткой – все узлы с одинаковыми метками принадлежат одному и тому же графу).

Параллельный алгоритм многократной маркировки перколяционных кластеров Хошена–Копельмана

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

Работу алгоритма можно разделить на три этапа.

1. Инициализация. Первый процесс загружает решетку в оперативную память из файла, преобразует ее по входным параметрам, распределяет узлы в группы по процессам и отсылает им. Остальные процессы ждут получения своей группы узлов. Получив такую группу, процессы выделяют из них подгруппу внешне связанных узлов, то есть узлов, связанных с узлами из групп других процессов. Для каждого узла своей группы задаются начальные значения меток в соответствии с абсолютным (в рамках всей решетки) номером узла. Каждый процесс создает группу внешних узлов, связанных с узлами своей группы, и инициализирует их метками связанных узлов своей группы.

2. Работа алгоритма. Каждый процесс запускает алгоритм ММК на своей группе узлов.

3. Обмен информацией. Происходит в несколько шагов до тех пор, пока после очередного обмена метки узлов всех процессов не перестанут изменяться. Обмен информацией

можно разделить на три этапа. Этап 1 – каждый процесс посылает значения меток узлов из подгруппы внешне связанных узлов тем процессам, с которыми эти узлы связаны. Этап 2 – каждый процесс принимает значения меток от процессов, имеющих узлы, связанные с узлами данного. Эти значения присваиваются меткам узлов из группы внешних узлов. Если хотя бы одна из меток узлов внешней группы была изменена, процесс должен повторить обмен информацией. Все метки узлов своей группы, равные замененным меткам внешней группы, также должны быть заменены. Этап 3 – отправка сообщения первому процессу о том, должен ли данный процесс повторить обмен информацией с другими процессами. Первый процесс получает от всех процессов подобную информацию. Если все процессы не нуждаются в повторном обмене, первый процесс рассылает всем остальным сигнал о завершении работы всего алгоритма и начинает процедуру сбора данных по меткам их узлов. Если же хотя бы один процесс сообщает, что ему необходимо продолжить обмен, всем процессам придется повторить процедуру обмена информацией.

Важным моментом в работе алгоритма является создание механизма линковки меток. Во время работы алгоритма ММК 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. Например, если данный узел имеет метку F, а его сосед метку Y, то ему и его соседу присваивается метка $\min(F, Y)$, но также надо заменить метки всех узлов решетки, имеющие значение F или Y на $\min(F, Y)$. Получается, что время работы алгоритма зависит от размера решетки N как $O(N^2)$. Механизм линковки меток позволяет добиться скорости $O(N)$. Суть этого механизма состоит в следующем.

Если пронумеровать все узлы решетки, им можно поставить в соответствие массив меток и массив ссылок, которые инициализируются номерами узлов. Массив ссылок – массив адресов в рамках массива. Например, если мы хотим узнать метку узла под номером 7, обращаемся в массив ссылок к элементу 7. Значение ссылки элемента 7 есть 7. Если значение ссылки в массиве ссылок равно номеру элемента этого массива, то по ссылке не надо никуда переходить, остается обратиться к тому же самому номеру элемента в массиве меток для получения значения метки, то есть обращаемся к элементу 7 массива ссылок и получаем значение метки 7.

В процессе изменения меток в алгоритме ММК изменяются не сами метки, а значения элементов массива ссылок. Например, если узел 2 связан с узлом 6 (рис. 1, шаг 2), то узлу 6 в массиве ссылок проставляется адрес на узел 2. Далее, если узел 1 связан с узлом 4, узлу 4 проставляется адрес на 1 (рис. 1, шаг 3). Аналогично, если узел 8 связан с узлом 6, то узлу 8 ставится адрес 6. После выполнения этих операций для получения метки узла 8 надо выполнить описанные выше действия: обращаемся к массиву ссылок по номеру 8, получаем адрес 6; обращаемся к номеру 6, получаем адрес 2; обращаемся к номеру 2, получаем адрес 2. Так как адрес совпал с номером элемента, надо обратиться в массив меток под номером 2. Получаем для узла 8 метку, равную 2.

После достаточно длительного процесса перелинковки следует приводить массив ссылок к такому виду, при котором количество переходов по адресам не превышает 1.

Алгоритм *многократной маркировки перколяционных кластеро*в (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 изуче-

Массив ссылок		Массив ссылок		Массив ссылок		Массив ссылок		Метки
1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3
4	4	4	4	4	1	4	1	4
5	5	5	5	5	5	5	5	5
6	6	6	2	6	2	6	2	6
7	7	7	4	7	4	7	1	7
8	8	8	8	8	6	8	2	8
Шаг 1		Шаг 2		Шаг 3		Шаг 4		

Рис. 1. Механизм линковки меток

Fig. 1. A mark labeling mechanism

ния их эффективности становится достаточно актуальной проблемой.

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, учитывая ее большой размер, логично проводить распараллеливание процесса 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 [5, 6].

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].

Описание разделов суперкомпьютера, на которых проводилось исследование

Программа маркировки кластеров Load, реализующая параллельный алгоритм ММПК, запускалась на пяти разделах (Broadwell, Cascadelake, Skylake, Optan, KNL) суперкомпьютера МВС-10П ОП, установленного в *Межведомственном суперкомпьютерном центре РАН* (МСЦ РАН) [10].

Алгоритм ММПК используется для изучения процессов распространения эпидемий как часть мультиагентной имитационной модели.

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

Исследование поведения алгоритма при 100- и 50-процентной загруженности узлов на всех указанных разделах, кроме раздела Optane (запущен в эксплуатацию в 2021 году), рассмотрено в [5]. Существенной экономии времени выполнения при частичной загрузке на большинстве систем не обнаружено.

Рассмотрим вычислительные характеристики разделов суперкомпьютера МВС-10П ОП подробнее.

МВС-10П ОП предоставляется пользователям МСЦ в режиме коллективного доступа к пяти разделам – Broadwell, Cascadelake, Skylake, Optan и KNL.

- 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.
- Cascadelake, 146 вычислительных модулей на базе процессоров Intel Xeon Platinum 8268 (Cascade Lake), 192 ГБ оперативной памяти на модуль, пиковая производительность модуля – 4,454 TFLOPS, 7 008 ядер в разделе. В 2021 году данный раздел был существенно увеличен – с 51 модуля до 146.
- 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.
- Optane, 6 вычислительных модулей на базе процессоров Intel(R) Xeon(R) Gold 6248R, 685 ГБ оперативной памяти на модуль, пиковая производительность модуля – 4,6 TFLOPS, 288 ядер в разделе. Память Intel Optane представляет собой системное решение для ускорения платформ, созданных на базе новых процессоров Intel Core седьмого поколения. В этом решении используется технология Intel Optane, основанная на носителе памяти 3D XPoint вместе с драйвером технологии хранения Intel Rapid (Intel RST). Этот новый носитель памяти находится между процессором и считающимися медленными устройствами хранения информации с SATA-интерфейсом (жесткий диск, гибридный жесткий диск или твердотельный накопитель SATA). При этом можно хранить часто используемые данные и программы ближе к процессору, чтобы система могла получать доступ к информации гораздо быстрее, что подразумевает общее повышение быстродействия системы.

- KNL, 22 вычислительных модуля на базе процессоров Intel Xeon Phi 7290, 96 ГБ оперативной памяти на модуль, пиковая производительность модуля – 3.456 TFLOPS, 1 584 ядра в разделе.

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

Анализ работы алгоритма на разделах суперкомпьютера

На рисунке 2 показан график зависимости времени работы программы Load от количества запрашиваемых процессорных ядер на разделе Broadwell суперкомпьютера МВС-10П ОП при 100-процентной загруженности вычислительных узлов.

Вычисления проводились на 64–1 216 процессорах с шагом 32 (по числу ядер в узле). Среднее время расчета составило 528 сек., минимальное время запуска – 353 сек. на 224 ядрах.

На рисунке 3 показан график зависимости времени работы программы Load от количества запрашиваемых процессорных ядер на разделе Cascadelake суперкомпьютера МВС-10П ОП при 100-процентной загруженности вычислительных узлов.

Вычисления проводились на 48–888 процессорах с шагом 24 (1/2 числа ядер в узле). Среднее время расчета составило 415 сек., минимальное время запуска – 327 сек. на 192 ядрах.

На рисунке 4 показан график зависимости времени работы программы Load от количества запрашиваемых процессорных ядер на разделе Skylake суперкомпьютера МВС-10П ОП при 100-процентной загруженности вычислительных узлов.

Вычисления проводились на 36–612 процессорах с шагом 36 (по числу ядер в узле). Среднее время расчета составило 398 сек., минимальное время запуска – 317 сек. на 108 ядрах.

На рисунке 5 показан график зависимости времени работы программы Load от количества запрашиваемых процессорных ядер на разделе Optane суперкомпьютера МВС-10П ОП при 100-процентной загруженности вычислительных узлов.

Рис. 2. Расчет на разделе Broadwell
Fig. 2. A calculation on the Broadwell

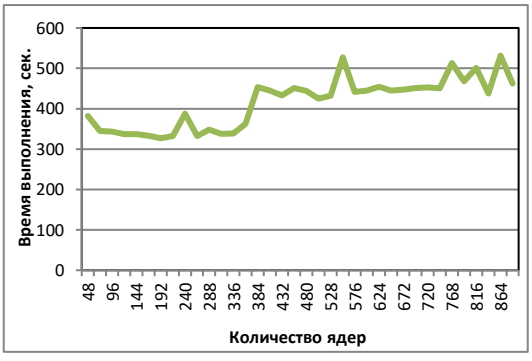

Рис. 3. Расчет на разделе Cascadelake

Fig. 3. A calculation on the Cascadelake

Рис. 4. Расчет на разделе Skylake

Fig. 4. A calculation on the Skylake

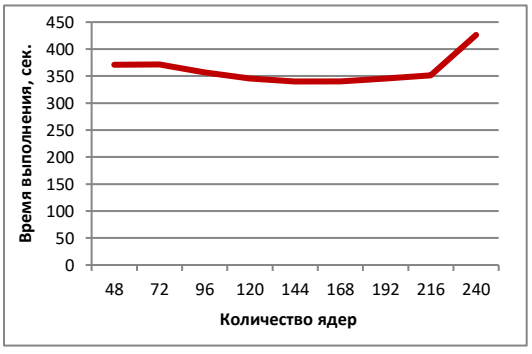

Рис. 5. Расчет на разделе Optane

Fig. 5. A calculation on the Optane

Вычисления проводились на 48–240 процессорах с шагом 24 (1/2 числа ядер в узле). Среднее время расчета составило 361 сек., минимальное время запуска – 340 сек. на 168 ядрах.

На рисунке 6 показан график зависимости времени работы программы Load от количества запрашиваемых процессорных ядер на разделе KNL суперкомпьютера МВС-10П ОП при 100-процентной загруженности вычислительных узлов.

Вычисления проводились на 36–752 процессорах с шагом 36 (по числу ядер в узле). Среднее время расчета составило 1 332 сек., минимальное время запуска – 1 203 сек. на 180 ядрах.

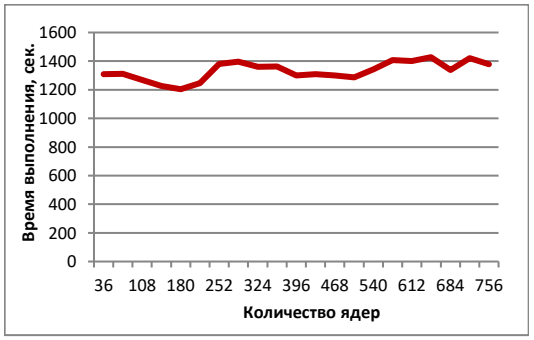

Рис. 6. Расчет на разделе KNL

Fig. 6. A calculation on the KNL

На рисунке 7 показан 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 раздел Skylake. Минимальное время счета достигается при использовании 100–200 ядер.

Минимальное время расчета показывает раздел Skylake. Максимальное – раздел KNL.

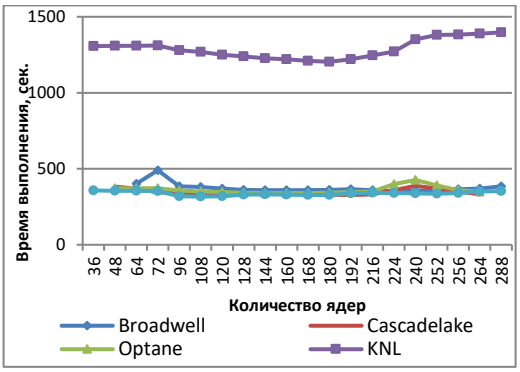

Рис. 7. Сводный график зависимости времени расчета от количества запрашиваемых процессорных ядер при 100-процентной загрузке вычислительных узлов

Fig. 7. A summary graph of the dependence of the computation time on the number of requested processor cores at 100 % load of computational nodes

Минимальное время счета достигается при использовании 100–200 ядер для всех разделов суперкомпьютера, из чего можно сделать вывод о неэффективности использования более чем 200 ядер при распараллеливании алгоритма ММПК.

Заключение

В статье сделан сравнительный анализ работы алгоритма ММПК на пяти различных разделах суперкомпьютера МВС-10П ОП при полной загрузке вычислительных узлов. Установлен диапазон оптимального количества запрашиваемых процессорных ядер, проведено ранжирование систем по времени выполнения численного эксперимента.

Расчеты проводились в МСЦ РАН на высокопроизводительной вычислительной системе МВС-10П ОП.

Работа выполнена в рамках государственного задания № 0580-2021-0014 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.: Едиториал УРСС, 2002. 112 с.

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

4. Утакаева И.Х. Имитационное моделирование распространения эпидемий на основе агентного подхода // Научный журнал КубГАУ. 2016. № 121. URL: http://ej.kubagro.ru/2016/07/pdf/85.pdf (дата обращения: 09.06.2021). DOI: 10.21515/1990-4665-121-085.

5. 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.

6. Lapshina S.Yu. The optimal processor cores' number research for the parallel cluster multiple labeling technique. Lobachevskii J. of Mathematics, 2020, vol. 41, pp. 2552–2557. DOI: 10.1134/s1995080220120240.

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

8. Savin G., Chipornyak A., Rybakov A., Shumilin S. Process mining: Realization and optimization of process discovery algorithm. Lobachevskii J. of Mathematics, 2020, vol. 41, no. 12, pp. 2566–2574. DOI: 10.1134/S199508022012032X.

9. Рыбаков А.А. Двухуровневое распараллеливание для оптимизации вычислений на суперкомпьютере при расчете задач газовой динамики // VI Всерос. конф. Вычислительный эксперимент в аэроакустике: сб. тезисов. 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 J. of Mathematics, vol. 40, no. 11, pp. 1853–1862. DOI: 10.1134/S1995080219110271.

Software & Systems Received 24.06.21
DOI: 10.15827/0236-235X.136.589-596 2021, vol. 34, no. 4, pp. 589–596

A comparative analysis of the parallel cluster multiple labeling technique on various sections of the MVS-10P OP supercomputer

S.Yu. Lapshina [1], *Senior Researcher, lapshina@jscc.ru*

[1] *Joint Supercomputer Center of the Russian Academy of Sciences – Branch of Federal State Institution "Scientific Research Institute for System Analysis of the Russian Academy of Sciences" (JSCC RAS – Branch of SRISA), Moscow, 119334, Russian Federation*

Abstract. The paper provides a comparative analysis of the Parallel Cluster Multiple Labeling Technique on five different sections of the MVS-10P OP supercomputer (taking into account the addition of a new section in 2021 and modernization of existing ones) installed at the JSCC RAS.

At the JSCC RAS, the Parallel Cluster Multiple Labeling Technique is used to study the processes of epidemic spread. At the same time, it is a versatile tool that can be used in any field as a tool for differentiating large lattice clusters receiving data as input in an application-independent format. There are known developments using this algorithm to study the processes of water flow through porous materials, the behavior of oil reservoirs, and the spread of forest fires.

The supercomputer simulation experiment involved the improved version of the technique for multiple labeling of Hoshen-Kopelman percolation clusters associated with the labels linking mechanism improved for using on a multiprocessor system.

The paper provides a comparative analysis of the execution time of the algorithm for multiple marking of Hoshen-Kopelman percolation clusters at full load of computing nodes and different values of input parameters on five partitions (Broadwell, Cascadelake, Skylake, Optan, KNL) of the MVS-10P OP supercomputer installed at the Interdepartmental Supercomputer Center of the Russian Academy of Sciences.

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

Acknowledgements. *The work has beed carried out at the JSCC RAS in the framework of state assignment no. 0580-2021-0014 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, 112 p. (in Russ.).

3. Kazakov S.A., Shebeko Yu.A. *Introduction to the Practice of Simulation and Analysis of Complex Processes and Systems Behavior*. Moscow, MIET Publ., 2006, 147 p. (in Russ.).

4. Utakaeva I.H. Simulation modeling of distribution of epidemics on the basis of agent approach. *Scientific J. of KubSAU*, 2016, no. 121. Available at: http://ej.kubagro.ru/2016/07/pdf/85.pdf (accessed June 9, 2021) (in Russ.). DOI: 10.21515/1990-4665-121-085.

5. 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 and Systems*, 2020, vol. 33, no. 4, pp. 557–563 (in Russ.). DOI: 10.15827/0236-235X.132.557-563.

6. Lapshina S.Yu. The optimal processor cores' number research for the parallel cluster multiple labeling technique. *Lobachevskii J. of Mathematics*, 2020, vol. 41, pp. 2552–2557. DOI: 10.1134/s1995080220120240.

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

8. Savin G., Chipornyak A., Rybakov A., Shumilin S. Process mining: Realization and optimization of process discovery algorithm. *Lobachevskii J. of Mathematics*, 2020, vol. 41, no. 12, pp. 2566–2574. DOI: 10.1134/S199508022012032X.

9. Rybakov A.A. Two-level parallelization for optimizing computations on a supercomputer when calculating gas dynamics problems. *Proc. 6th All-Russ. 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 J. of Mathematics*, vol. 40, no. 11, pp. 1853–1862. DOI: 10.1134/S1995080219110271.

Для цитирования

Лапшина С.Ю. Сравнительный анализ работы алгоритма многократной маркировки перколяционных кластеров на различных разделах суперкомпьютера МВС-10П ОП // Программные продукты и системы. 2021. Т. 34. № 4. С. 589–596. DOI: 10.15827/0236-235X.136.589-596.

For citation

Lapshina S.Yu. A comparative analysis of the parallel cluster multiple labeling technique on various sections of the MVS-10P OP supercomputer. *Software & Systems*, 2021, vol. 34, no. 4, pp. 589–596 (in Russ.). DOI: 10.15827/0236-235X.136.589-596.