УДК 004

Воробьев М. Ю., Рыбаков А. А., Никсон Муганда Очара
1,2. МСЦ РАН – филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия
3. University of Venda, Thohoyandou Limpopo, South Africa
1. nordmike@jscc.ru, 2.rybakov@jscc.ru, 3.nixon.muganda@gmail.com

ИССЛЕДОВАНИЕ МАСШТАБИРУЕМОСТИ ПЛОТНЫХ ПАРАЛЛЕЛЬНЫ ВЫЧИСЛЕНИЙ НА МИКРОПРОЦЕССОРАХ INTEL[2]

В настоящее время особое значение в мире приобрели технологии искусственного интеллекта и обработки больших данных. Объемы накапливаемой информации постоянно возрастают, и для обработки данных требуются все более мощные вычислительные ресурсы. При этом особую значимость приобретают методы распределенной обработки данных с высокой степенью параллельности. В то же время количество ядер (и потоков) в современных микропроцессорах также возрастает, и, как следствие, возникает вопрос об эффективности их использования при организации параллельных вычислений по обработке данных. В статье анализируются современные микропроцессоры Intel (вычислительные узлы на их основе), которыми оборудованы вычислительные системы МСЦ РАН, с точки зрения масштабируемости плотных вычислений. Исследование основано на оригинальных методиках измерения эффективности масшибирования высоконагруженных вычислений для систем с общей памятью, данные методики разработаны в МСЦ РАН.

***КЛЮЧЕВЫЕ СЛОВА**:* обработка данных, параллельные вычисления, высоконагруженные вычисления, масштабируемость вычислений, эффективность.

Введение

Основным показателем эффективности микропроцессора в плане выполнения вычислительной задачи является время выполнения. Сегодня в ускорении выполнения задачи основную роль играет не тактовая частота микропроцессора, а параллельность исполнения. Любая более или менее серьезная вычислительная задача разрабатывается с учетом современных параллельных аппаратных архитектур, при этом внимание уделяется обеспечению параллельности исполнения на различных логических уровнях: распределенное выполнение между удаленными расчетными серверами, параллельное исполнение задачи между узлами суперкомпьютерного кластера [1], многопоточное выполнение в рамках систем с общей памятью [2], векторизация − низкоуровневое распараллеливание выполнения задачи на уровне отдельных инструкций [3]. При этом, для каждого уровня распараллеливания возникает вопрос об эффективности распараллеливания, то есть насколько увеличение степени параллельности приводит к ускорение счета. Увеличение степени параллельности может подразумевать как

[2] Статья выполнена при поддержке РФФИ, грант № 19-57-60004\20

увеличение количества серверов или вычислительных узлов, так и увеличение количества ядер в отдельных микропроцессорах, а также увеличение ширины векторных регистров, используемых в векторных инструкциях. Данный показатель эффективности распараллеливания будем называть показателем масштабируемости приложения. Конечно для каждой вычислительной задачи данный коэффициент будет свой. Введем его более формальное описание. Пусть существует некоторая вычислительная задача, а также есть возможность запуска этой задачи с разной степенью параллельности вычислений от *1* до *n*. В данной статье исследуется эффективность масштабирования в рамках одного вычислительного узла с общей памятью, так что в данном случае число *n* означает максимальное количество потоков, которые могут быть запущены на рассматриваемом узле. Пусть также для каждого количества потоков *i* от *1* до *n* известно время выполнения приложения *t(i)*. Ускорение от распараллеливания приложения на *i* потоков определяется естественным образом и вычисляется по формуле (1):

$$s(i) = t(1) / t(i) \quad (1)$$

Также определим показатель эффективности масштабирования вычислений, который вычисляется по формуле (2):

$$e(i) = s(i) / i \quad (2)$$

Будем считать, что в случае идеального распараллеливания задачи при увеличении количества потоков в *i* раз время выполнения задачи сокращается также в *i* раз, в этом случае $s(i) = i$, а $e(i) = 1$. В этом смысле при идеальном распараллеливании задачи можно говорить о линейной масштабируемости с эффективностью, равной единице. На самом деле бывают случаи даже сверхлинейной масштабируемости [4], когда эффективность превышает единицу, но это скорее исключение, чем ожидаемый эффект.

В данной статье будет приведено сравнение эффективности масштабирования расчетных приложений решения задач газовой динамики для вычислительных узлов на базе микропроцессоров Intel, которые входят в состав разделов суперкомпьютеров МСЦ РАН [5].

Расчет эффективности масштабируемости вычислений

Для анализа эффективности масштабируемости вычислений была выбрана задача нахождения точного решения при распаде произвольного разрыва (которая также называется задачей Римана) [6]. Данная задача в различных вариациях (в частности приближенные решения) широко используется при реализации численных методов решения задач газовой динамики (в частности в

реализации вычислительных методов Годунова) [7, 8, 9]. Привлекательностью расчетного ядра точного решения задачи Римана является естественный вид его программного контекста: реализация содержит многочисленные особенности, в том числе разветвленное управление, команды переходов, вложенные циклы с нерегулярным числом итераций, вычисление трансцендентных функций. И в то же время вычислительное ядро обладает достаточно компактной формой, что позволяет детально оценить возможности по оптимизации и использовать все особенности целевой архитектуры. В данной статье задача Римана о распаде произвольного разрыва используется в двух вариантах реализации: скалярная реализация, а также векторная реализация с использованием набора инструкций AVX-512 [10,11], которая позволяет одновременно решать 16 экземпляров данной задачи в одном вычислительном потоке микропроцессора. Реализация векторной версии римановского решателя является отдельной темой, которая раскрыта в работах [12,13], в контексте данной статьи стоит лишь упомянуть, что рассматриваемая векторная реализация позволяет достичь ускорения римановского решателя примерно в 7 раз по сравнению со скалярной версией на вещественных числах с одинарной точностью. Используемые в статье реализации римановского решателя доступны в сети Интернет [14].

Таблица 1. Конфигурация вычислительных узлов, на которых проводились замеры показателей масштабирования вычислений.

Семейство микропроцессоров	Количество процессоров / ядер / потоков в узле	Объем оперативной памяти в узле	Поддержка AVX-512
Xeon Haswell	2 / 28 / 56	128 Гб	нет
Xeon Broadwell	2 / 32 / 64	128 Гб	нет
Xeon Phi KNL	1 / 72 / 288	96 Гб	да
Xeon Skylake	2 / 36 / 72	192 Гб	да
Xeon Cascade Lake	2 / 48 / 96	192 Гб	да

Для прогона римановского решателя были собраны данные решения задачи на общеизвестных тестовых сценариях, таких как задача Сода, задача Лакса, задача Эйнфельдта, задача Вудворда-Колелла, задача Шу-Ошера, и других [15], обеспечивающих хорошее покрытие программного кода решателя. Для тестирования были использованы обе версии римановского решателя: скалярная и векторная. Из вычислительных узлов были рассмотрены все современные узлы, входящие в состав суперкомпьютеров МСЦ РАН (таблица 1). Данные вычислительные узлы базируются на микропроцессорах Intel: Xeon Haswell, Xeon Broadwell, Xeon Phi

KNL, Xeon Skylake, Xeon Cascade Lake (из них три последних поддерживают набор векторных инструкций AVX-512). Для скалярной версии римановского решателя сравнение проводилось для всех представленных вычислительных узлов, для векторизованной версии римановского решателя сравнивались узлы на базе микропроцессоров Xeon Phi KNL, Xeon Skylake, Xeon Cascade Lake.

Результаты сравнения эффективности масштабирования вычислений

На рис. 1 представлены графики ускорения скалярной версии римановского решателя при увеличении количества потоков с 1 до 160. Для каждого вычислительного узла явно просматриваются отрезки квазилинейного ускорения, которые завершаются заметными провалами. Длина этих отрезков во всех случаях равняется суммарному количеству ядер в узле, а провалы обусловлены конфликтами за аппаратные ресурсы. При этом стоит отметить, что хоть максимальное количество доступных потоков для вычислительного узла на базе микропроцессора KNL равняется 288, однако на графике не показаны значения больше 160, так как сверх этого значения наблюдается только деградация производительности, и наиболее эффективное использование зафиксировано в диапазоне потоков 140-144. Также можно отметить, что для всех микропроцессоров ускорение близко к линейному до тех пор, пока каждый поток запущен на своем отдельном ядре.

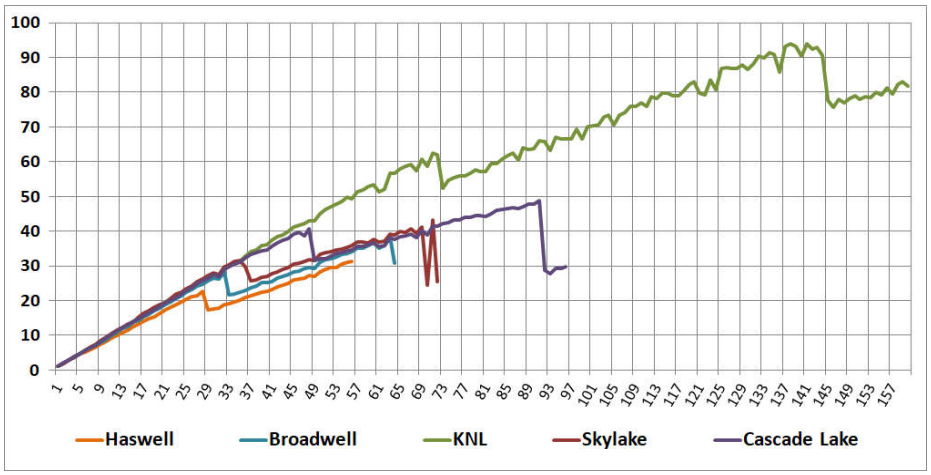

Рис. 3. График ускорения скалярной версии римановского решателя для микропроцессоров Haswell, Broadwell, KNL, Skylake, Cascade Lake для количества потоков от 1 до 160.

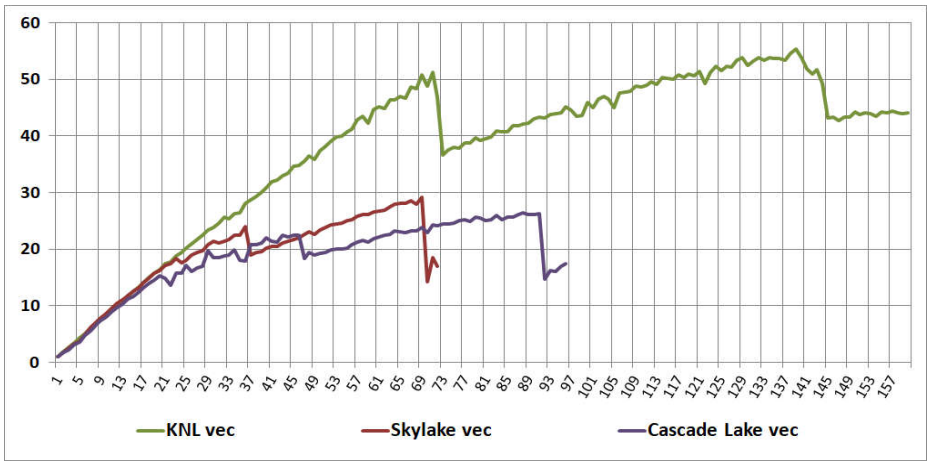

Рис. 2. График ускорения векторизованной версии римановского решателя для микропроцессоров KNL, Skylake, Cascade Lake для количества потоков от 1 до 160.

На рис. 2 представлены аналогичные данные, но уже для векторизованной версии римановского решателя (соответственно только для микропроцессоров KNL, Skylake, Cascade Lake). Из графиков видно, что характер ускорения для KNL практически не поменялся, тогда как Skylake и Cascade Lake демонстрируют серьезную деградацию ускорения вычислений уже на небольшом количестве потоков.

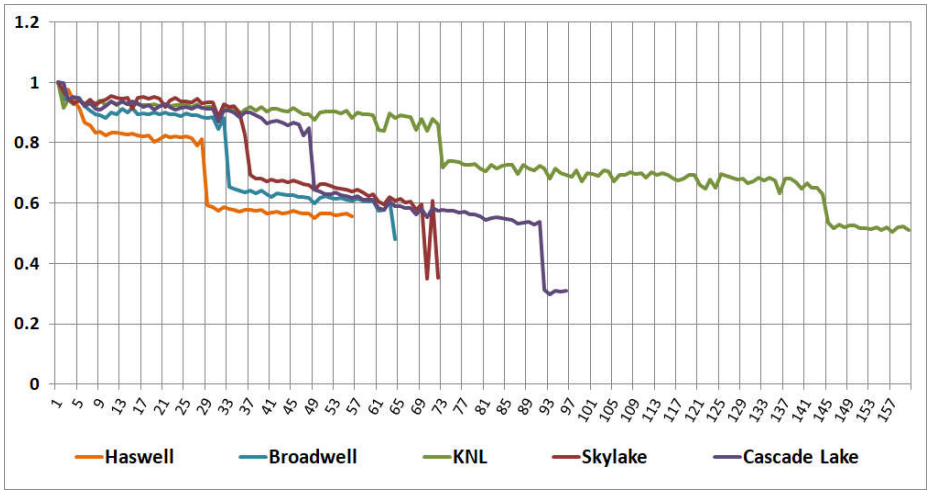

Рис. 3. График масштабируемости скалярной версии римановского решателя для микропроцессоров Haswell, Broadwell, KNL, Skylake, Cascade Lake для количества потоков от 1 до 160.

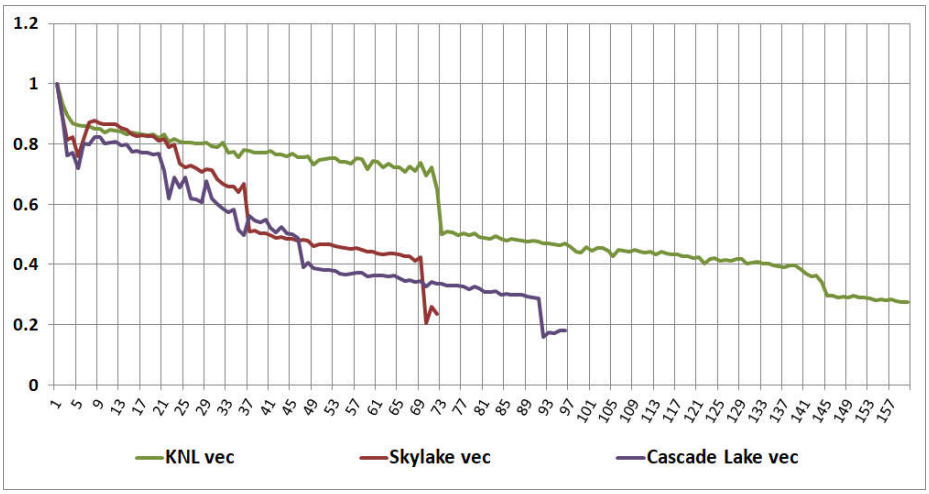

Рис. 4. График масштабируемости векторизованной версии римановского решателя для микропроцессоров KNL, Skylake, Cascade Lake для количества потоков от 1 до 160.

На рис. 3 и рис. 4 приведены данные показателей эффективности масштабирования римановского решателя (скалярной и векторизованной версии соответственно). Данные иллюстрации более информативные. Например, из рис. 3 видно, что масштабируемость при количестве потоков, не превышающем количество ядер вычислительного узла, находится для всех микропроцессоров на хорошем уровне в диапазоне 0,8 - 1,0. Рис. 4 наглядно демонстрирует хорошую масштабируемость векторизованного римановского решателя для микропроцессора KNL с показателем, близким к 0,8 вплоть до использования 72 потоков, тогда как узлы на базе микропроцессоров Skylake и Cascade Lake деградируют на масштабировании векторизованной версии римановского решателя на достаточно небольшом количестве потоков.

Заключение

В работе отмечена актуальность задачи организации распределенных и параллельных вычислений в эпоху возрастающих потребностей по применению искусственного интеллекта и обработки больших объемов данных. Для эффективного использования вычислительных ресурсов при проведении параллельных вычислений особую важность приобретает показатель эффективности масштабирования вычислений. В работе выполнен сравнительный анализ эффективности масштабирования типовой расчетной задачи (задача Римана о распаде произвольного разрыва) в скалярной и векторной постановке для вычислительных узлов, входящих в состав суперкомпьютеров МСЦ РАН. На основании выполненных расчетов был сделан вывод, что скалярная версия

решателя одинаково эффективно масштабируется на узлах любого типа при количестве потоков, не превышающем количество физических ядер узла. Также отмечен тот факт, что при использовании плотных векторных вычислений лучшие показатели масштабируемости достигнуты на микропроцессорах KNL [16].

Публикация выполнена в рамках проекта РФФИ «Модели анализа «больших данных» в задачах планирования устойчивого регионального развития: открытый инновационный подход» (№ 19-57-60004). В работе был использован суперкомпьютер МВС-10П ОП, находящийся в МСЦ РАН.

ЛИТЕРАТУРА:

1. Klenk B., Froning H. An overview of MPI characteristics of exascale proxy applications. J. M. Kunkel et al. (Eds.): ISC High Performance 2017, LNCS, Vol. 10266, pp. 217-236 (2017).
2. Dorris J., Kurzak J., Luszczek P. Task-based Cholesky decomposition on Knights Corner using OpenMP. M. Taufer et al. (Eds.): ISC High Performance Workshop 2016, LNCS, Vol. 9945, pp. 544-562 (2016).
3. Shabanov B. M., Rybakov A. A., Shumilin S. S. Vectorization of high performance scientific calculations using AVX-512 instruction set. Lobachevskii Journal of Mathematics, Vol. 40, No. 5, pp. 580-598 (2019).
4. Бендерский Л. А., Любимов Д. А., Рыбаков А. А. Анализ эффективности масштабирования при расчетах высокоскоростных турбулентных течений на суперкомпьютере RANS/ILES методом высокого разрешения. Труды НИИСИ РАН, Т. 7, № 4, сс. 32-40 (2017).
5. JSCC RAS Supercomputing resources, http://www.jscc.ru/supercomputing-resources, last accessed 2020/11/01.
6. Куликовский А. Г., Погорелов Н. В., Семенов А. Ю. Математические вопросы численного решения гиперболических систем уравнений. ФИЗМАТЛИТ, М., 608 с. (2001).
7. Годунов С. К., Забродин А. В., Иванов М. Я., Крайко А. Н., Прокопов Г. П. Численное решение многомерных задач газовой динамики. Наука, М., 400 с. (1976).
8. Shumlak U., Udrea B. An approximate Riemann solver for MHD computations on parallel architectures. Proceedings of the 15th AIAA Computational Fluid Dynamics Conference (2001).
9. Ferreira C. R., Mandli K. T., Bader M. Vectorization of Riemann solvers for the single- and multi-layer shallow water equations. Proceedings of the 2018 International Conference on High Performance Computing and Simulations, HPCS 2018, pp. 415-422 (2018).
10. Intel 64 and IA-32 architectures software developer's manual. Combined volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D and 4. Intel Corporation (2019).
11. Intel Intrinsics Guide, https://software.intel.com/sites/landingpage/IntrinsicsGuide, last accesed 2020/11/01.
12. Rybakov A. A., Shumilin S. S. Vectorization of the Riemann solver using the AVX-512 instruction set. Program Systems: Theory and Applications, Vol. 10, № 3 (42), pp. 41-58 (2019).
13. Рыбаков А. А., Шумилин С. С. Исследование эффективности векторизации гнезд циклов с нерегулярным числом итераций. Программные системы: Теория и алгоритмы, Т. 10, № 4 (43), сс. 77-96 (2019).

14. riemann_vec, https://github.com/r-aax/riemann_vec/releases/tag/metric, last accessed 2020/10/01.
15. Булат П. В., Волков К. Н. Одномерные задачи газовой динамики и их решение при помощи разностных схем высокой разрешающей способности. Научно-технический вестник информационных технологий, механики и оптики, 15:4, сс. 731-740 (2015).
16. Jeffers J., Reinders J., Sodani A. Intel Xeon Phi processor high performance programming, Knights Landing edition. Morgan Kaufmann (2016).

УДК 004

Данилин А. О. Ибатулин М. Ю.
1. МГТУ «СТАНКИН»
2. МГТУ «СТАНКИН» uits_stankin@mail.ru

ПРИМЕНЕНИЕ КОМПЕТЕНТНОСТНОЙ МОДЕЛИ ОБУЧЕНИЯ НА ОСНОВЕ ТЕХНОЛОГИЙ ЦИФРОВОГО СЛЕДА

В работе рассматривается важность перехода к компетентностной модели обучения. Представлен метод построения цифрового профиля компетенции в рамках «Университета 20-35» и его результаты. Предлагается подход к реализации цифрового профиля студента в рамках МГТУ «СТАНКИН».

КЛЮЧЕВЫЕ СЛОВА: Цифровой след, университет 20-35, компетенции, профиль студента, индивидуальные траектории

Введение

В настоящее время в России происходит трансформация всех областей деятельности человека и сфера образования не является исключением. Классическая или «знаниевая» форма образования подразумевает строгое разделение дисциплин и критериев оценки по этим дисциплинам. В течение семестра обучающийся должен запоминать то, что ему преподают на занятиях, а во время сессии показать полученный уровень знаний по этому предмету. Также на результаты экзамена может оказать влияние эмоционально-физическое состояние обучающегося или экзаменатора, также могут оказать влияние какие-либо личностные конфликты, что в значительной мере искажает фактический уровень знаний обучающегося.

Одним из вариантов решения обозначенной проблемы является бально-рейтинговая система, основанная на принципе накопления итоговой оценки. Так во время учебного процесса обучающемуся выставляются баллы, которые опираются на все его достижения: посещение лекций, практических занятий, активность на занятиях, сроках выполнения заданий. Этот подход позволяет обучающимся самому решать на чем сосредоточить свои усилия в данный момент, в зависимости от текущих результатов. Но при реализации данного подхода зачастую появляются трудности из-за того, что приходится изменять как критерии оценки, так и сами учебные платы, опираясь на новые шаблоны. Кроме того, переход занимает достаточно