



Современные серверные ARM-процессоры для суперЭВМ: A64FX и другие Начальные данные тестов производительности

Михаил Борисович Кузьминский[✉]

Институт органической химии им. Н.Д. Зелинского РАН, Москва, Россия
kus@free.net[✉] (подробнее об авторе на с. 129)

Аннотация. Дан сравнительный анализ производительности серверных ARM-процессоров, используемых на суперЭВМ или ориентированных в частности на высокопроизводительные вычисления (HPC). В стартовый анализ производительности были отобраны Fujitsu A64FX, Marvell ThunderX2 и Huawei Kunpeng 920. Обзор производительности для HPC сосредоточен в первую очередь на тестах и приложениях для A64FX, поддерживающего более длинные, чем у других ARM-процессоров, вектора и имеющего большую пиковую производительность. Производительность A64FX сопоставлена с соответствующими данными для Intel Xeon Skylake и Cascade Lake, и AMD EPYC с Zen 2 и 3 (Roma и Milan), а также с GPU Nvidia V100 и A100. Сформулирован краткий набор потенциальных плюсов и минусов микроархитектуры A64FX. Сопоставлены данные о производительности, получаемой с применением различных компиляторов для A64FX. Сформулированы признаки, когда A64FX дает обычно преимущества в производительности относительно x86-64, а когда — проигрывает x86-64.

Подтверждается, что применение A64FX в суперЭВМ может расти далее. Возможно, гегемония x86-64 в HPC будет уменьшаться, в том числе за счет расширения применения серверных ARM-процессоров. Однако проведенный анализ A64FX и ожидаемых в ближайшее время новых процессоров архитектуры AArch64 показал, что ведущим в этом процессе не обязательно окажется A64FX.

Ключевые слова и фразы: ARM, AArch64, A64FX, x86-64, высокопроизводительные вычисления, суперЭВМ, тесты производительности

Для цитирования: Кузьминский М. Б. *Современные серверные ARM-процессоры для суперЭВМ: A64FX и другие. Начальные данные тестов производительности* // Программные системы: теория и приложения. 2022. Т. 13, № 1(52). С. 63–129. http://psta.psiras.ru/read/psta2022_1_63-129.pdf

© Кузьминский М. Б.

2022 ©

This Article in English:

http://psta.psiras.ru/read/psta2022_1_139-204.pdf

Введение

Рост доли рынка систем для НРС, занимаемой ARM-процессорами [1] сопровождается изменениями последнего времени на рынке соответствующих аппаратных средств, гораздо более радикальными, чем рост доли рынка серверных x86-64 процессоров, занимаемого AMD [1]. Задержки с развитием полупроводниковой технологии у Intel [2] оспариваются в [1] утверждением, что 10 нм у Intel по размерам практически эквивалентно 7 нм у TSMC. Замедление действия закона Мура (см., например, [1, 3–7]) заставляет обращать особое внимание на электропитание (см., например, [4–7]). Актуальность энергоэффективности (производительность на Ватт) давно учитывается в квантовой химии с особым вниманием к ARM-процессорам [8] и с сопоставлением 64-разрядных ARM с x86-64 [9]. Для суперЭВМ энергоэффективность особенно важна [1].

Область НРС и суперЭВМ естественно становятся лидерами предполагаемых изменений. Соответственно возрастает и актуальность обзоров по всем используемым в НРС средствам вычислительной техники. Отставание Европы в данной области пытаются преодолеть новая инициатива *EPI* (*European Processor Initiative*)^(URL) по разработке и производству в том числе новых ARM-процессоров и новая инфраструктура *PRACE* (*Pan-European High Performance Computing Infrastructure and Services*)^(URL), где широко анализируются разные новейшие аппаратные средства для НРС и данные тестов производительности [1, 2, 10]. В аналогичном отечественном обзоре [11] рассмотрены и отечественные аппаратные средства.

Серверные ARM-процессоры или вычислительные системы на их базе начали производиться в разных регионах мира, включая Китай [12], а соответствующие суперЭВМ предлагают Cray/HPE и Fujitsu [13].

Средства высокопроизводительной вычислительной техники продолжают быстро развиваться. Оперативно появляются новые данные о производительности используемого программного обеспечения, ценные для оптимального выбора и для построения новых суперЭВМ. Искусственный интеллект требует нетрадиционных для НРС типов данных (двойной точности, DP) [13]. Новые серверные процессоры AArch64 начали оттеснять x86-64 с их лидирующей позиции для НРС, и, в частности, суперЭВМ благодаря высокой энергоэффективности ARM-процессоров [11–14]. Активное продвижение RISC-V [15], высокая

пиковая производительность IBM Power10 (претендующего и на лидерство по DP-производительности и предполагаемого к выпуску в 2021 году) [16] и применение RISC-процессора SW26010 в по-прежнему находящейся в десятке лидеров TOP500 китайской суперЭВМ Sunway TaihuLight (см., например, [17, 18]) позволяют говорить и о некоем возрождении RISC.

Перечисленное определяет высокую актуальность приводимого ниже обзора данных о производительности в HPC (и, в частности, на суперЭВМ) серверных ARM-процессоров (в первую очередь наиболее высокопроизводительного Fujitsu A64FX).

1. Аппаратные особенности процессоров AArch64, важные для HPC

1.1. Серверные процессоры AArch64 с поддержкой 128-битной векторизации

Первым широко известным AArch64-процессором, который стал использоваться в суперЭВМ из TOP500, стал, видимо, Marvell ThunderX2 [19], в котором поддерживается работа со 128-битными векторами (Advanced SIMD, NEON). Хотя и ранее в суперЭВМ использовались другие ARM-процессоры [20]. Различные данные о производительности ThunderX2 уже хорошо известны. В нем не очень высокая стоимость сочетается с конкурентоспособностью с массовыми серверными x86-64 процессорами по производительности для HPC [21]. Появились *неподтвержденные данные*^{URL} и о более высоких показателях производительности ThunderX2, например, относительно Xeon Skylake 6148 — в том числе по SPECrate2017_int_peak. Однако Xeon Skylake по производительности ThunderX2 обычно опережают, в том числе на тестах *SPEC OMP2012*^{URL} или, например, в различных приложениях вычислительной химии [21] *или*^{URL}. По данным [7], ThunderX2 может опережать Xeon Skylake на HPC-приложениях, интенсивно работающих с памятью, давая при этом большую энергоэффективность, а более вычислительно интенсивные приложения быстрее считать на Skylake. Классической областью HPC-приложений, в которых так важна пропускная способность памяти, является вычислительная гидродинамика (CFD) [20], а вычислительная химия включает, например, молекулярную динамику (МД), относимую к вычислительно интенсивной области HPC.

Таблица 1 сопоставляет основные показатели производительности процессоров ThunderX2 с характеристиками новых AArch64-процессоров. За исключением поддерживающего SVE [22] Fujitsu A64FX, остальные указанные в таблице 1 процессоры ARM поддерживают такую же, как ThunderX2, 128-битную векторизацию. Число операций с плавающей запятой за такт на ядро Xeon Scalable всех поколений выше, чем у всех этих процессоров; соответственно (пропорционально тактовой частоте) у ядер Xeon выше и пиковая производительность (FLOPS). А другие AArch64 процессоры имеют более высокую по сравнению с ThunderX2 пиковую производительность (по умолчанию мы говорим о работе с DP, стандартной для традиционных HPC-приложений). Новые публикации о производительности ThunderX2 продолжают появляться (см., например, [23]). Соответствующие результаты в большей степени важны для использования вычислительных систем на базе ThunderX2, поскольку производительность у более новых доступных процессоров AArch64 обычно выше, и они актуальнее для приобретения. Поэтому, учитывая много опубликованных статей о производительности ThunderX2, она рассматривается в данном обзоре только при сопоставлениях с другими более новыми серверными ARM-процессорами.

Последующие предполагавшиеся к выпуску модели процессора ThunhderX3 [24, 25] по всем важнейшим параметрам, включая количество и частоту ядер, могли превосходить ThunderX2 (см. таблицу 1). Хотя в ThunderX3 не предполагалось использовать SVE, поскольку разработчики опасались пока еще недостаточной ее отработанности для программных средств, зато число векторных NEON-устройств было удвоено относительно ThunderX2, и резко возросшая пиковая производительность превосходила бы и A64FX. Перейти на работу с SVE планировалось в ThunderX4. Однако в середине 2020 года Marvell сделала заявление, которое можно интерпретировать как отказ от выпуска ThunderX3 и переход на работу в отличных от HPC направлениях. В качестве другого примера доступного серверного 7 нм-процессора ARM следует указать 64-ядерный Huawei Kunpeng 920 [12], который изначально предполагалось использовать для HPC, и в [2, 10, 11] относят к таким процессорам. Он использует более современную, чем в ThunderX2, архитектуру ARM v8.2-A (как и в A64FX, но без SVE-расширения). В нем поддерживаются SIMD-блоки NEON, как и в ThunderX2. Kunpeng 920 превосходит его по числу ядер и другим показателям (см. таблицу 1), в том числе и по пиковой

Таблица 1. Основные характеристики ARM-процессоров для HPC

Параметры	A64FX	Kunpeng 920-6426	Thunder X2CN9980	Thunder X3	Altra (Q80)	Graviton2
Пиковая производительность (DP), TFLOPS	2,8-3,4	1,33	До 0,64	До 4,75	2,12	
Технология	TSMC 7 нм	TSMC 7 нм	TSMC 14 нм	TSMC 7 нм	TSMC 7 нм	TSMC 7 нм
Число ядер	48+4	64	32	До 96	80	64
SMP	1(ccNUMA)	До 4 (NUMA)	2/4(NUMA)	1-2 (NUMA)	1-2 (NUMA)	1
Тактовая частота	1,8-2,2 ГГц ¹	2,6 ГГц	2,1-2,5 ГГц ¹	До 3,1 ГГц	2,6-3,3 ГГц	2,5 ГГц
Многопоточность	нет	нет	SMT0/2/4	SMT4	нет	нет
Векторизация, бит	SVE,512	NEON,128	NEON,128 (2xFMA)	NEON,128 (4xFMA)	NEON,128 (2 устройства)	NEON,128
Архитектура	ARM v8.2-A +SVE	ARM v8.2-A (Neoverse N1)	ARM v8.1 Vulcan	ARM v8.3+ Triton	ARM v8.2+ (Neoverse N1)	Neoverse N1
Кэш L1 I+D	3 МБ (64 КБx48)	64КБ+64КБ на ядро	32+32 КБ на ядро	64+32 КБ на ядро	64+64 КБ на ядро	4+4 МБ
Кэш L2	32 МБ (4x8МБ)	32 МБ (512 КБ на ядро)	256 КБ на ядро	512 КБ на ядро	1 МБ на ядро	64 МБ
Кэш L3	нет	64 МБ (1 МБ на ядро)	32 МБ (1МБ на ядро)	90 МБ	До 32 МБ	32 МБ
Тип памяти	HBM2	8xDDR4-2933	8xDDR4-2666	8xDDR4-3200	8xDDR4-3200	8xDDR4-3200
Пропускная способность	1 ТБ/с	188 ГБ/с	171 ГБ/с	205 ГБ/с	205 ГБ/с	205 ГБ/с
Емкость памяти	32 ГБ	До 2 ТБ	До 2 ТБ x2 ²	Нет данных	До 1 ТБ	До 1 ТБ
Интерфейс I/O	PCIe-v3 x16 +TofuD	PCIe-v4 40x (до x16)	PCIe-v3 48x, 56x	16 x PCIe-v4 x4	PCIe-v4 128x	PCIe-v4 64x
TDP, Вт	Нет данных	180	180/2,2 ГГц	100-240	250 ³	80-110

¹максимум в boost-режиме;²для двухпроцессорного сервера;³для 3,3 ГГц.

Данные таблицы взяты с сайтов производителей.

производительности. Из аппаратных особенностей Kunpeng 920 следует указать необычную по сравнению с процессорами Xeon иерархию памяти, включая NUMA у двух суперкластеров ядер (SCCL) внутри чипа Kunpeng (см., например, [14, 26]), а также возможность поддержки ccNUMA в масштабах чиплетов на их базе [12]. Применяемое в Kunpeng образование групп процессорных ядер в чем-то похоже на используемые в A64FX и Zen, и это, вероятно, связано с общим для серверных процессоров существенном росте числа содержащихся в них ядер. Однако публикаций по HPC-тестам производительности Kunpeng 920 пока немного.

Наиболее массовыми сейчас следует считать данные тестов SPECcpu 2017 [27]. Соответствующие данные и сопоставление с серверами на базе современных x86-64 процессоров приведены в таблице 2.

ТАБЛИЦА 2. Сравнительные данные тестов производительности SPECrate 2017 base для серверов с Kunpeng 920 и процессорами x86-64

	Integer	Floating point
Двухпроцессорные		
Kunpeng 920-7260	318	263
Xeon Skylake 8180	297	276
Xeon Cascade Lake 8280	303	308
EPYC Rome, 7742	701	524
EPYC Milan, 7763	839	651
Четырехпроцессорные		
Kunpeng 920-7260	628	516
Xeon Skylake 8180	564	523
Xeon Cascade Lake 8280	670	566

По этим данным видно, что 64-ядерный Kunpeng 920 7260 с частотой 2,6 ГГц, выигрывая немного на целочисленной производительности, уступает старшим моделям Xeon Skylake в расчетах с плавающей запятой.

Приведены максимальные на 18.09.2021 данные с указанными процессорами. Данные более современных Xeon Ice Lake здесь не приводятся, так как они еще быстрее, чем Kunpeng 920.

Поскольку для традиционных HPC-приложений важна работа с плавающей запятой, далее по умолчанию мы говорим именно о ней. Более современные Cascade Lake увеличивают отрыв по производительности от Kunpeng 920, а Xeon Scalable третьего поколения еще быстрее.

SPEScpu 2017 нечасто используется для оценок производительности для НРС, тем более для суперЭВМ. Для Kunpeng 920 имеются данные по тестам пропускной способности памяти (stream) [10], см. (таблицу 3).

ТАБЛИЦА 3. Сопоставление данных тестов stream (ГБ/с) в двухпроцессорных узлах с Kunpeng 920, ThunderX2 и процессорами x86-64; Все пропускные способности — в ГБ/с¹

Процессор	Copy	Scale	Add	Triad
Kunpeng 920-6426	322	322	324	324
ThunderX2/2,5 ГГц	~ 225	~ 225	~ 240	~ 240
EPYC Rome 7742	338	338	340	340
Xeon 8160/2,1 ГГц	~ 180	~ 180	~ 200	~ 200
Xeon 8174/2,4 ГГц	~ 180	~ 180	~ 200	~ 200

¹Приведены наивысшие достигнутые в [10] результаты.

Эти показатели важны для НРС-приложений, в которых работа с памятью лимитирует поддерживаемую производительность (например, для CFD [20]). Kunpeng 920 опережают здесь ThunderX2 и Xeon Skylake, и уступают AMD EPYC Roma — что соответствует поддерживаемой этими процессорами памяти (см. выше таблицу 1) и ниже таблицу 4).

В RoCE (100 Гб/с)-кластере из 4 двухпроцессорных серверов с Kunpeng 920 были проведены тесты HPCG с MPI-распараллеливанием; данные для 1–4 узлов приведены на рисунке 1 [10], основанном

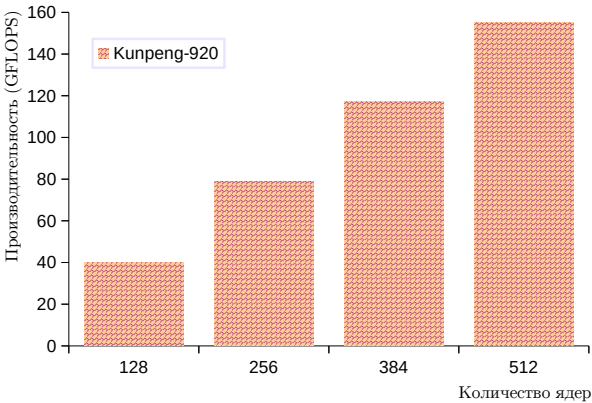


РИСУНОК 1. Производительность Kunpeng 920 в HPCG на рисунке 7 в [10]. Однопроцессорный сервер с ThunderX2/2,5 ГГц

Таблица 4. Основные параметры A64FX и современных x86-64 процессоров

Параметры	A64FX	Xeon Platinum 8180	Xeon Platinum 8280	Xeon Platinum 8380	AMD EPYC 7763 (milan)	AMD EPYC 7662 (ROME)
Архитектура	ARM-v8.2-A + SVE	Skylake	Cascade Lake	Ice Lake (Sunny Cove)	Zen 3	Zen 2
Технология	TSMC 7 нм	14 нм	14 нм	10 нм	TSMC 7 нм	TSMC 7 нм
Число ядер	48+4	28	28 (32 ¹)	40	64	64
Частота, ГГц	1,8-2,2	2,5/3,8 ²	2,7/4,0 ²	2,3/3,4 ²	2,45/3,5 ²	2,0/3,3 ²
SMP сокетов	1	через UPI до 8	через UPI до 8 ¹	До 8	1/2	1/2
Многопоточность	нет	SMT2	SMT2	SMT2	SMT2	SMT2
Векторизация, бит	SVE,512	AVX-512	AVX-512	AVX-512	AVX-256 (AVX2)	AVX-256 (AVX2)
FLOPS/такт (DP, на 1 ядро)	32	32	32	32	16	16
Кэш L1 (I+D)	64КБ+64КБ на ядро 4-way	32КБ+32КБ на ядро 8-way	32КБ+32КБ на ядро 8-way	32КБ+48КБ на ядро 8/12-way	32КБ+32КБ на ядро 8-way	32КБ+32КБ на ядро 8-way
Кэш L2	32МБ (8МБ×4) 16-way	1 МБ/ядро 16-way	1 МБ/ядро 16-way	512 КБ/ядро 8-way	32 МБ 8-way	32 МБ 8-way
Кэш L3	нет	38,5 МБ 11-way	38,5 МБ 11-way	60 МБ 16-way	256 МБ 16-way	256 МБ 16-way
Память	HBM2	6xDDR4-2666	6xDDR4-2933	8xDDR4-3200	8xDDR4-3200	8xDDR4-3200
	32 ГБ	До 768 ГБ	До 1 ТБ	До 6 ТБ	До 4 ТБ	До 4 ТБ
Пиковая пропускная способность ³	1 ТБ/с (4x256 ГБ/с)	128 ГБ/с	141 ГБ/с	205 ГБ/с	205 ГБ/с	205 ГБ/с
I/O	PCIe-v3×16 + TofuD	PCIe-v3 48x	PCIe-v3 48x	PCIe-v4 64x	PCIe-v4 128x	PCIe-v4 128x
TDP, Вт	Нет данных	205	205	270	280	225
Цена, \$	Нет данных	10010	10010	8100	7890	7000

¹Для Cascade Lake AP 92XX — 2 сокета; ²Повышенные значения — максимальная турбо-частота; ³ пиковая величина. Данные процессоров взяты с сайтов [Intel](#)^[url] и [AMD](#)^[url] 5.09.2021. Использованы также европейские данные [PRACE](#)^[url].

в этом тесте получил заметно меньше 20 GFLOPS, а двухпроцессорный сервер с Kunpeng — около 40 GFLOPS [10], что свидетельствует о его более высокой производительности, чем у ThunderX2. Двухпроцессорный сервер с Kunpeng 920 в тесте HPCG можно сопоставить с двухпроцессорными серверами на базе 24-ядерных Xeon Skylake: с Xeon Platinum 8160/2,7 ГГц производительность заметно ниже 40 GFLOPS, а с Xeon Platinum 8174/2,7 ГГц — около 40 GFLOPS [10]. Таким образом, двухпроцессорные серверы с Kunpeng 920 оказались конкурентоспособными по производительности в HPCG с двухпроцессорными серверами с Xeon Skylake, имеющим поддержку AVX-512. Это связано с тем, что в HPCG не так важна работа с векторами/матрицами, как в HPL.

Хорошо известно, что работа с матрицами, в первую очередь умножение матриц (GEMM), часто лимитирует время расчета. В качестве примеров можно указать, например, HPL [28] или квантовохимические расчеты по учитывающим корреляцию методам [6]. Имеется немало поддерживающих BLAS математических библиотек [29]. Однако они очень быстро развиваются, и появляются также новые библиотеки, в том числе и для задач оптимизации на новых компьютерных архитектурах. Сравнительный анализ различных библиотек с BLAS для DGEMM-вычислений в двухпроцессорном сервере с 48-ядерными Kunpeng 920/2,6 ГГц показал достижение максимальной производительности в OpenBLAS, где затем была оптимизирована DGEMM за счет явного учета особенностей работы с NUMA, что дало выигрыш при работе с матрицами большого размера. Максимальная производительность была достигнута с использованием для распараллеливания в OpenBLAS средств pthreads, а не OpenMP. Достигнутый благодаря переработке DGEMM прирост производительности составляет порядка 20% [14].

Другая знаменитая для HPC вещь — умножение разреженной матрицы на вектор, SpMV. Оно актуально, например, при численном решении уравнений в частных производных [30]. В квантовой химии оно используется, например, для расчетов методом конфигурационного взаимодействия [31]. В [32] проведено исследование на Kunpeng 920 достигаемой производительности при использовании SIMD-команд NEON с анализом зависимости от формата хранения данных в матрице. Следует также отметить, что для производительности SpMV найден важным и явный учет наличия NUMA-архитектуры [30].

В [33] проведено подробное сопоставление производительности Kunpeng 920-6426 и 18-ядерных Xeon Gold 6140. В случаях, когда производительность ограничивается работой с кэшем или памятью, Kunpeng оказывался впереди. Но в более вычислительно-интенсивных тестах с использованием векторизации Xeon выигрывал по производительности. *Исследование в НИВЦ МГУ в рамках совместного проекта с Huawei*^[URL] охватывает сравнение Kunpeng 920-6426 не только с Xeon 6140, но и с EPYC 7763 (Milan, с Zen 3). Kunpeng 920 показал более высокую производительность, чем Xeon 6140, в вычислительно-интенсивных тестах, где не используется векторизация, и оказался медленнее в работе с разреженными матрицами. Процессор EPYC 7763 обычно обходил по производительности и Xeon 6140, и Kunpeng 920. При этом некоторые показатели EPYC 7763 близки к Kunpeng 920: по 64 ядра с сопоставимыми частотами, в обоих поддержка только 128-битных векторов, по 8 каналов памяти (хоть в EPYC поддерживается DDR4-3200 против DDR4-2933 в Kunpeng). Эти данные носят отчасти предварительный характер.

В общем можно считать ясным отставание Kunpeng 920 по производительности по сравнению со старшими моделями современных серверных процессоров Intel Xeon Scalable 2-го или 3-го поколений, и AMD EPYC с архитектурой Zen 2 или 3. Появились и более новые модели серверных процессоров Kunpeng, в том числе Kunpeng 920 7265, в которых ядра работают на более высокой частоте 3,0 ГГц (*список моделей доступен по ссылке*^[URL]). Подробнее данные о производительности Kunpeng 920 здесь не анализируются по причине небольшого числа соответствующих публикаций. А в 2021 году ожидался выпуск Kunpeng 930 (5 нм) с поддержкой SVE, который может стать более используемым в суперЭВМ. Однако реализация этого сейчас, возможно, заморожена санкциями по отношению к Huawei, в том числе ограничениями поставки современных микросхем со стороны TSMC.

Другим современным AArch64 процессором, который мог бы представлять интерес для НРС, является *80-ядерный Ampere Altra Q80*^[URL] [1] (см. таблицу 1) и более новый *128-ядерный Altra Max*^[URL]. Они имеют больше ядер, чем 64-ядерный Kunpeng, и более высокую пиковую производительность. Документация от Ampere утверждает, что их 80-ядерный процессор имеет производительность, сопоставимую с AMD Roma EPYC 7742, и в 2 раза выше, чем у Intel Cascade Lake SP Xeon Platinum 8280 [1] (общую информацию о процессорах такого

типа см. выше, таблицу 4). Так, *на тесте молекулярной динамики*^{URL} с применением NAMD процессор Altra Q80/3,3 ГГц оказался быстрее 1–2 процессорных серверов с Xeon 8280, и однопроцессорного — с EYUC 7742, но уступил двухпроцессорному серверу с EYUC 7742. Однако в Xeon при этом AVX-512 не использовано, что может понизить производительность Xeon скажем, в 2 раза. А информация о распараллеливании вообще не была приведена. Эти данные получены для тестов SPEC (в том числе SPECrate2017_int и _fp). Данные о других более надежных тестах производительности, актуальных для HPC, у этих процессоров отсутствуют. Производитель вообще ориентирован не на традиционный HPC-рынок, а скорее на ИИ и другие области применения, и Altra в обзоре не рассматривается.

По всем приводимым в данном обзоре параметрам (см. выше таблицу 1), важным для производительности, 64-ядерный Amazon AWS Graviton2 *уступает*^{URL} Ampere Altra, как и в вышеуказанных SPECcpu2017 тестах, и также ориентирован не на традиционные HPC, а на применение облачных технологий, и в обзоре не рассматривается.

С учетом всего сказанного выше про поддерживающие NEON-векторизацию процессоры AArch64 более подробно анализ их производительности в данном обзоре не проводится.

1.2. Процессоры с поддержкой SVE: Fujitsu A64FX и вычислительные системы на его базе

Кроме A64FX, поддерживающие SVE процессоры еще только планируются к выпуску в ближайшее время. Это — Huawei Kunpeng 930, в котором планировалась также поддержка многопоточности (SMT), и европейский SiPearl Rhea [1]. В нем предполагается поддержка SVE-256 бит (не только с DP, но и с bfloat16), но и дополнительная к HBM2E память DDR5. Возможно, это отражает стремление разработчиков к немного более массовым применениям для HPC, поскольку актуальность сегодняшней аппаратной поддержки для GEMM оспаривается [4].

1.2.1. Особенности важных для производительности аппаратных средств A64FX

Рассмотренные выше ARM-процессоры на момент их выпуска отличались повышенной по отношению к Xeon-процессорам пропускной

способностью памяти, которая стала более актуальной для НРС в период перехода на работу с многоядерными процессорами, поскольку пропускная способность не так быстро увеличивалась [20]. Как было отмечено выше, это очень важно, например, для CFD-расчетов.

Процессор A64FX стал знаменит в НРС не только благодаря рекордной на сегодня пиковой производительности, но и благодаря стабильному лидерству в TOP500 японской суперЭВМ Fugaku, построенной на этих процессорах вообще без применения GPU [13]. Он отличается от самых современных x86-64 процессоров не только применением кардинально более быстрой памяти HBM2 (см. таблицу 4), но и построением всей иерархии памяти (хотя в EPYC Zen 2 и 3 есть аналогии в построении кэша) и продвинутой SVE-системой работы с векторами. Для управления энергопотреблением в A64FX имеются также более тонкие, чем в Хеон, средства, позволяющие влиять на энергоэффективность. Для оптимизации производительности и энергоэффективности все это нужно учитывать в компиляторах, библиотеках для НРС, а иногда и собственно в приложениях.

Наиболее важные для производительности показатели A64FX приведены выше (таблица 4), где они сопоставлены с различными моделями современных серверных процессоров Intel и AMD. На рисунке 2 в соответствии с [35, рис. 2] представлено общее построение A64FX, иллюстрирующее и построение памяти A64FX [34, 35]. Этот процессор

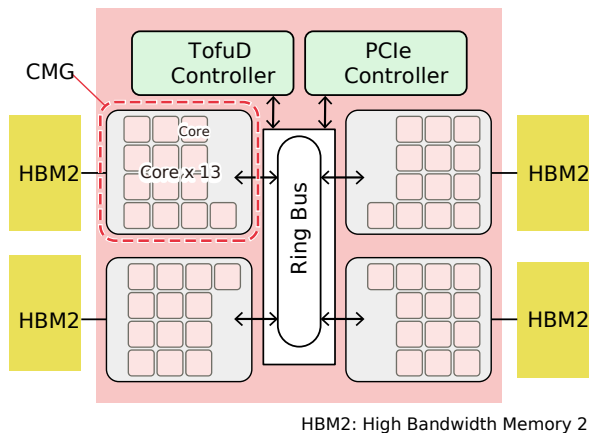


Рисунок 2. Общая архитектура A64FX

состоит из 4 групп ядер и памяти (CMG, кластера ядер) — так, что каждый CMG имеет по 12 ядер (плюс одно вспомогательное), контроллер памяти HBM2 и разделяет кэш L2 (а каждое ядро имеет свой кэш L1). Все CMG в процессоре объединены единой кольцевой шиной, так что кэш L2 и вся HBM2 являются общими для всего A64FX, а процессор является ccNUMA-системой [35]. Такое построение иерархии памяти отличается от всех современных серверных процессоров x86-64, в которых общим является кэш L3, а кэши L1 и L2 принадлежат ядрам (см. выше, таблицу 4). В A64FX размер области кэша L1 в микросхеме составляет более 10% от каждого ядра, и добавление еще одного уровня кэша в ядре заметно увеличило бы размер кристалла (die), что было найдено экономически нецелесообразным [34]. При этом используемая в A64FX HBM2 имеет кардинально более высокую пропускную способность, чем DDR4 у процессоров x86-64 — по 256 ГБ/с на каждый CMG (см. выше таблицу 4).

Каждый CMG содержит кроме 12 вычислительных ядер еще одно вспомогательное — для обслуживания операционной системы. Однако в серверах суперЭВМ Fujitsu PRIMEHPC FX700 *используются модели A64FX без таких вспомогательных ядер*^{URL}.

Поддержка SMT в A64FX отсутствует. Это может иметь экономический характер (учитывая, что из-за этого в A64FX отказались и от дополнительного уровня кэша на ядрах, и от добавления к HBM2 еще и DDR4-памяти [34]) — но, возможно, сделано потому, что SMT далеко не всегда полезна для HPC (см. например [10]), а реализация такой сложной вещи требует не только дополнительной площади, но и энергии. Рост производительности из-за применения SMT не всегда наблюдается в HPC, поскольку производительность большинства приложений ограничена чаще или тем, что разные потоки используют одни и те же исполнительные блоки, или пропускной способностью памяти [10]. В [10] демонстрируются случаи, когда отключение SMT в процессоре AArch64 дает рост производительности, хотя есть и приложения, где SMT важна.

Важнейшим добавлением A64FX относительно ARM v8.2-A является *SVE (Scalable Vector Extension)*^{URL}. В A64FX возможна теперь работа с векторами длиной 128, 256 и 512 бит. В современных процессорах с векторными операциями длина векторов указывается в битах, в том числе поскольку в ISA предполагаются векторные операции над разными типами данных, не только DP и одинарной точности (SP), но

и половинной точности, в том числе и `bfloat16`. SVE предполагает возможность работы с векторами разной длины от 128 до 2048 бит с шагом 128 [36], а программу для работы с аппаратной поддержкой векторов другой длины перетранслировать не надо (длина определяется уже во время выполнения). Это является одним из преимуществ SVE над AVX-512. Другим преимуществом SVE является использование там регистров предикатов (например, для полностью векторной обработки операций над массивами длины, не кратной аппаратной длине вектора) [36]. Один бит регистра предиката соответствует одному байту SVE-регистра, и это позволяет работать с фрагментами векторов.

Еще одним важным плюсом является поддержка векторных операций сборки/разборки (`gather/scatter`), иллюстрируемых двумя следующими операторами

$$\begin{aligned}\text{gather} : a[i] &= b[\text{index}[i]], \\ \text{scatter} : a[\text{index}[i]] &= b[i].\end{aligned}$$

В результате SVE способствует улучшению автовекторизации компиляторами.

Благодаря тому, что с SVE в A64FX можно работать с 512-битными векторами, за такт там можно получить 32 результата (DP) на ядро, как и в Xeon, начиная со Skylake. Однако ядер в A64FX гораздо больше, что дает ему более высокую пиковую производительность. SVE дает не только повышение производительности, но и снижение энергопотребления [37].

Еще одной важной особенно для суперЭВМ уникальностью A64FX является интеграция в нем управления RDMA-межсоединением TofuD. Эта сеть имеет топологию 6-мерной решетки/тора, использовавшейся ранее в суперЭВМ K, предшественнике Fugaku. Но пропускная способность TofuD из-за его интеграции в процессоре зависит теперь от его частоты. Для убыстренного (см. ниже) режима с частотой 2,2 ГГц она составляет 6,8 ГБ/с на каждый из 6 каналов связи [10, 34, 35]. Это меньше, чем в Infiniband HDR 4x и 16x, но для больших вычислительных систем со сложной топологией связи между узлами важна суммарная пропускная способность всех одновременно работающих каналов — 40,8 ГБ/с, что выше пропускной способности одной платы HDR 4x, но ниже HDR 12x. Задержка RDMA-коммуникации (Put 8 байт) в TofuD составляет 0.49–0.54 мкс [38], а в *Infiniband HDR (Nvidia/Mellanox ConnectX-6)*^{URL} — 0.6 мкс.

TofuD имеет в A64FX общий с PCIe-v3 контроллер, так что весь A64FX можно отнести к системе-на-кристалле (SoC).

В A64FX имеется специальный регулятор мощности. Имеются 2 основных режима выбора частоты — нормальный и убыстренный; в FX1000 этому соответствуют частоты 2,0 и 2,2 ГГц. В каждом из этих режимов есть возможность включить еще эконо-режим, в котором функционирующим остается только один из двух конвейеров АЛУ с плавающей запятой [34, 35]. Соответственно получается 4 режима управления энергопитанием, которые можно выбирать для достижения максимальной производительности или подбора энергоэффективности (это будет рассмотрено ниже). Считается, что GPU дают большую энергоэффективность по сравнению с процессорами. Однако в [35] энергоэффективность A64FX отмечена как сопоставимая с вычислительными системами на базе GPU.

Мы не рассматриваем в обзоре доступные данные о микроархитектуре A64FX; более детальные данные об ОоО, механизму предсказания переходов, построению TLB и другим вещам можно найти в *руководстве*^{URL}. Но у A64FX есть еще одна важнейшая для НРС особенность. Он разрабатывался именно для задач НРС [34], а в более узком плане — для создания суперЭВМ Fugaku [35]. В [34] указано на применение совместного проектирования (co-design) для A64FX, когда определяющие для производительности параметры будущего A64FX подбирались совместно с разработчиками программного обеспечения для НРС, и с учетом конкретных приложений. Для этого использовался эмулятор процессора. Например, для выбора оптимального числа ядер в SMG проверялась производительность HPL. В качестве одного из «участвовавших в отборе» приложений был и известный японский комплекс квантовохимических программ NTChem. В результате в [34] четко демонстрируется, почему Fujitsu были выбраны именно такие параметры. Несомненно, такой подход является очень привлекательным для ориентированных на НРС процессоров.

1.2.2. Потенциальные недостатки A64FX для производительности НРС-задач

В процессоре A64FX есть и ряд недостатков, обычно рассматриваемых относительно серверных процессоров x86-64 (см. выше, таблицу 4), которые часто вызваны стремлением не увеличивать стоимость (например, не такой большой фиксированный объем памяти

HBM2, к которой не добавляется DDR4) [34]. Имеются также данные 2017 года, что в ресурсах суперЭВМ NSF для HPC 86% заданий не требовали наличия больше 32 ГБ памяти в узлах [39]. А задержки HBM2 больше чем у DDR4 [35].

Следует отметить, что измеряемые в тестах величины задержки памяти включают несколько компонентов, и задержка в шине передачи данных может быть не основным вкладом в общую задержку. Хотя про существенно большие задержки HBM относительно DRAM известно достаточно давно, и на это указывается в ряде публикаций, количественных оценок не так много, а получаемые величины задержек зависят от работающих с памятью микросхем. В качестве примера укажем полученные в тесте Shuhai задержки при попадании на страницу (то есть когда она уже открыта) при работе с микросхемой ПЛИС на плате Xilinx Alveo U280 — 107 нс для HBM и 73 нс для DDR4 [40].

В качестве другого недостатка можно считать отсутствие кэша L3, причина чего объяснена выше.

При конструировании в A64FX внеочередного выполнения команд (OoO) также использовалось его моделирование с использованием небольших частей приложений, но решение о нужном количестве ресурсов для OoO в то время было затруднено недостаточной эффективностью нового компилятора. При этом ограничение ресурсов для OoO способствует снижению энергопотребления, а прогресс в работе компилятора может вообще ослабить проблемы с OoO [34].

В A64FX емкость буфера переупорядочивания (ROB) составляет 128 записей — существенно меньше, чем у современных x86-64 процессоров: в Xeon Skylake их 224, в Ice Lake — 352, в ожидаемом Alder Lake планируется 512; в EPYC Zen 2 — 224, в Zen 3 — 256. Число физических регистров (кроме отсутствующих в Xeon регистров предикатов) в A64FX также меньше, чем в Xeon Skylake [41]. Большая задержка времен выполнения команд может приводить к недостатку OoO [34]. А в ряде работ [42]–[45] отмечались большие задержки выполнения разных команд A64FX. Так что ограничения производительности A64FX из-за недостатков с OoO достаточно вероятны.

PCIe, используемая в A64FX, — это не последняя применяемая версия (см. выше, таблицу 4); SSD могут работать с PCIe-v4, а в ожидаемом Power10 будет уже PCIe-v5 [16]. При этом SSD используются и для HPC [46]. Но для A64FX важнее интеграция в процессоре контроллера с TofuID, а проблема высокоскоростного соединения

с GPU не стояла, поскольку их в суперЭВМ на A64FX применять не предполагается. Но пиковая производительность GPU Nvidia V100 и A100 в разы выше, чем у A64FX: 7,8 и 9,7 TFLOPS соответственно [47], а в новом AMD Instinct MI100 еще больше (11,5 TFLOPS) — против 3,4 TFLOPS в A64FX, и применение GPU сейчас актуально по крайней мере для уже эффективно работающих с GPU НРС-приложений. Из-за ограничений PCIe-v3 в A64FX в новой японской суперЭВМ Wisteria/BDEC-01 планируется использовать гетерогенную структуру с одними узлами на A64FX, а с другими — на Xeon Ice Lake с A100.

Наконец, надо иметь в виду, что бывают и НРС-тесты/приложения где применение SMT (не поддерживаемая в A64FX, как отмечено выше) может повысить производительность.

В нижеследующем разделе 3 данного обзора A64FX показывается и влияние отмеченных в 1.2.2 показателей A64FX на производительность.

1.2.3. О вычислительных системах на базе A64FX

Первое, что является необычным для давно использующих в кластерах для НРС двухпроцессорные серверы — что серверы с A64FX однопроцессорные, и производительность A64FX часто сопоставляется с двумя процессорами x86-64, к тому же на двух процессорах Xeon нередко бывает близкое к A64FX число ядер. Fujitsu с A64FX предлагает 2 суперкомпьютерные платформы PRIMEHPC — FX700 и FX1000. Последняя используется в суперЭВМ Fugaku, и обладает максимальными возможностями — до 384 узлов на стойку с водяным охлаждением, соединенных через TofuD. A64FX в узлах имеют частоту 2,2 ГГц и 48 ядер плюс 4 вспомогательных (см. рисунок 2). FX700 базируются на 2U-устройствах; на шасси помещается до 8 узлов, и работают с воздушным охлаждением. A64FX в них работает на частотах 1,8 или 2 ГГц, а вспомогательных ядер нет. Узлы связаны через *Infiniband EDR*^{URL}.

Следует обратить внимание на отличие A64FX в FX700 и в FX1000. В A64FX из FX700 нет поддержки TofuD, кластеры работают с Infiniband EDR или HDR100 — для этого используется разъем PCIe-v3 x16. Но зато есть еще PCIe-v3 x4, к которому подсоединяется SSD. Другие вычислительные системы с A64FX предлагает HPE/Cray — это Apollo 80 (ранее анонсированная как Cray CS500). Узлы здесь соединяются через Infiniband EDR или HDR100. Стойка 42U *может включать до 168 серверов с 48-ядерными A64FX с частотами 1,8 или 2 ГГц*^{URL} — аналогичными применяемым в FX700.

В целом следует обратить внимание на то, что в разных моделях A64FX отличается не только тактовая частота, но могут быть или не быть вспомогательные процессоры, а TofuD может не поддерживаться и будет работать с разной пропускной способностью в зависимости от частоты процессора. А все только что перечисленные вычислительные системы с A64FX не являются традиционными серверами, а исходно ориентируются на работу в кластерах для HPC.

2. A64FX: операционные системы и средства разработки программ для HPC

Для работы с A64FX можно использовать классические для HPC дистрибутивы Linux — RHEL 8/CentOS, и SLES 15/OpenSuSE. Здесь надо обратить внимание на необходимость их реальной установки на всех SSD всех вычислительных узлов кластеров — для хранения там корневой файловой системы. Вариант с хранением в этих узлах только резидентных в памяти образов системы на суперЭВМ Ookami (на специальных узлах для компиляции и отладки SSD там стояли исходно) затребовал на это порядка 14 Гбайт памяти в каждом узле — из-за работы со страницами емкостью 64 КБ, и столько «отнимает» даже супермаленький файл [39]. И поскольку Fugaku можно уже отнести к суперЭВМ EFLOPS-уровня (при работе с половинной точностью), для реализации HPC с таким сверхвысоким уровнем распараллеливания, усложненной иерархией памяти и другими сложностями дополнительно к Linux (RHEL) используется микроядро McKernel [48], которое и отрабатывает массовые системные вызовы для HPC, а более сложные переправляя в Linux.

Для A64FX доступно пять способных создавать коды с SVE компиляторов C/C++/Fortran и ряд математических библиотек. Имеется утверждение, что производительность большинства HPC-приложений сегодня ограничивается пропускной способностью памяти [49], что дает A64FX однозначное преимущество среди доступных процессоров. В A64FX, однако, отношение производительности к пропускной способности памяти другое, и реальные данные по HPC-приложениям и тесты производительности показывают, что A64FX далеко не всегда лидирует по производительности, и качество средств разработки программ крайне важно. В данном обзоре компиляторы и библиотеки анализируются в основном с точки зрения получаемой потребителем

производительности — а не для анализа возможных причин их недоработок по производительности и возможностей устранения этого. Также актуальный для выбора компилятора учет доступности в них современных версий C/C++/Fortran, в том числе средств OpenMP здесь не проводится.

Публикаций, демонстрирующих уровень эффективности работы компиляторов, уже немало. Но поскольку A64FX использует целый ряд оригинальных микроархитектурных решений, и стал доступным достаточно недавно, компиляторы оценивают как еще недостаточно «созревшие» [50], и хотя все современные их версии уже могут осуществлять автовекторизацию в SVE, в [39, 51] сделан, по сути, такой же вывод. Перечислим используемые в ряде публикаций доступные версии компиляторов: Fujitsu 4.5.0 (имеется два его режима — традиционный (FJtrad) и на базе LLVM (в том числе FJclang), где back-end базируется на LLVM 7; LLVM v.12 (там можно использовать векторизацию в средствах оптимизации циклов polly); GNU v.10.2.0, а также компиляторы ARM и HPE/Cray — недоступны на Fugaku из-за отсутствия лицензий (по крайней мере, во времена отправления используемых в обзоре публикаций). В качестве «незрелости» компиляторов для A64FX сошлемся для примера на данные об ошибках компилятора GNU, ошибки flang в LLVM [50], ошибки времени выполнения в компиляторах ARM и Cray [39, 52].

Использованный в [50, 53, 54] LLVM 12 — это современная стабильная версия, лишь в октябре 2021 года появилась версия 13.0.0. Непрерывное быстрое развитие других компиляторов для A64FX также ярко проявилось и в 2021 году, например, вышли новые версии ARM компилятора (в том числе 21.1) и Cray (12.0).

Компиляторы Cray и Fujitsu являются коммерческими, как и ARM компиляторы, входящие в ориентированный на HPC набор Allinea Studio, включающий в том числе компиляторы C/C++/Fortran и *Arm Performance Library*, *armPL*^{URL} (хотя имеется и его пробный бесплатный вариант), а свободно доступные компиляторы — GNU и LLVM. Основные опции всех этих компиляторов, предназначенные для включения векторизации, OpenMP и других видов оптимизации, приведены в [55].

ARM-компиляторы для HPC в качестве front-end используют cland/flang, а back-end основан на LLVM (в 21.1 — на Clang 11), где используются 2 блока векторизации: один естественный для циклов, а

другой дает параллелизм на уровне сверхслова (SLP), когда вместо набора из похожих независимых инструкций генерируются векторные инструкции [55]. Нужно также отметить реализованное впервые в ARM-компиляторе расширение C, ACLE (Arm C Language Extensions), которое было сделано и для SVE, и помогало автовекторизации в начальный период работы с SVE, а теперь устарело и планируется к удалению в следующих версиях компилятора.

Преимуществом компиляторов Fujitsu является поддержка некоторых повышающих производительность аппаратных особенностей A64FX, о которых не было сказано выше — так как они носят более тонкий характер, могли использоваться только в этих компиляторах, и их влияние почти не исследовалось. Однако в [43] изучено влияние FJtrad 4.4.0a на использование этих средств посредством опций, переменных окружения или директив pragma. Это — нулевое заполнение (позволяет улучшить работу с памятью, и в [43] показано ускорение работы в тесте stream triad, но аналогичные механизмы имеются и у других процессоров); применение аппаратных барьеров для ускоренной синхронизации нитей и разбиение кэша на секторы разного размера. Последнее позволяет лучше управлять пространством кэша, выделенным для каждой структуры данных в коде, и может помочь избежать «загрязнения» всего кэша. В [43] было показано увеличение за счет этого производительности при умножении плотной матрицы на вектор. А применение аппаратных барьеров дало ускорение синхронизации нитей в FJtrad (fcc), в том числе и по сравнению с программной ее реализацией в gcc 10.2.

В одном из самых широких исследований компиляторов [50] анализировался не только выбор опций компиляторов, но также эффективность рекомендуемого Fujitsu выбора числа нитей OpenMP и потоков MPI при гибридном распараллеливании на A64FX.

Fujitsu и RIKEN (японский физико-химический институт, где установлена суперЭВМ Fugaku) рекомендуют использовать на каждом CMG по 12 нитей OpenMP, а поверх — распараллеливание с MPI ранга 4 (по числу CMG). В [50] использовались и другие варианты выбора комбинаций OpenMP/MPI.

Для микроядер от приоритетных для RIKEN приложений лучшую производительность почти всегда давал FJtrad; GNU в нескольких случаях генерировал более быстрые коды, но в некоторых —

ошибки. В группе миниприложений от RIKEN, например, для квантовохимической программы NTChem, GNU обычно уступает другим компиляторам. В тестах stream рост производительности до 51% относительно FJtrad давали LLVM и GNU. В тестах SPECspeed 2017 с многонитевыми рабочими нагрузками с плавающей запятой, и в SPECcomp GNU показывал обычно самые плохие результаты. Для тестов SPEC на C/C++ компиляторы на основе LLVM (включая FJclang) и в некоторых случаях GNU могут дать преимущество во время выполнения по сравнению с FJtrad, однако для Fortran LLVM не дает существенного выигрыша в производительности. В тестах *Polyhedral Benchmark* (*PolyBench*)^{[URL](#)}, которые включают 30 однопоточных микроядер, LLVM с *polly* по данным [50] очень часто кардинально опережал другие компиляторы, но иногда и существенно уступал. Но надо иметь в виду, что PolyBench и применяется для тестирования производительности при разработке средств *polly*^{[URL](#)}. А в [50] эти средства вообще отнесены к мало полезным где-либо вне самого набора тестов PolyBench.

В [56] в рамках теста BP3 из CEED (Center for Efficient Exascale Discretizations) gcc 10.2.0 дал более высокую производительность, чем Fujitsu (fcc) 4.4.0a.

В [57] на A64FX/1,8 ГГц тестирование производительности (пропускной способности) DAXPY как в однопоточном варианте, так и на всем процессоре с распараллеливанием в OpenMP или pthreads показало, что gcc 11 опережал по производительности ARMclang 21. В качестве недостатка gcc 11 была указано только отсутствие векторизации элементарных функций в glibc (см. об этом ниже).

Однако в [53] на приложении квантовой хромодинамики QPACE 4, и на тестах stream и умножения матриц в кластере на FX700 с A64FX/1,8 ГГц в качестве оптимальных по производительности оказались gcc 10.1 и 10.2, а gcc 11.1 и LLVM 12 отставали. Неясно также, насколько на этих результатах сказалось применение ACLE. В [54] на этой же вычислительной системе, с этими же тестами и приложением было добавлено сопоставление и с другими компиляторами, в том числе с ARMclang 21.0 и FJtrad, FJclang 4.3.1 — но все компиляторы, кроме gcc 10.1/10.2, дали плохую производительность (в том числе и gcc 11.1).

В [50] был сделан вывод, что система компиляторов для НРС с A64FX еще не настолько развита, как для x86, и нельзя указать на один в основном самый лучший компилятор, в разных типах

тестов/приложений может быть разная рекомендация. Представляется, что исследованная в [50, 53, 54] версия GNU 10.2 все-таки не должна рекомендоваться к применению из-за немалого числа ошибок; теперь следует первоначально ориентироваться на более современную стабильную версию GNU, а 10.2 не поддерживается. Поддержка A64FX и улучшения работы с SVE декларированы в поддерживаемой ныне версии 10.3, а в стабильной версии 11.2 исправлен ряд ошибок, обеспечивается частичная поддержка уже OpenMP 5.0, и автовекторизация при работе с комплексными числами — эти версии GNU и выглядят рекомендуемыми сейчас для широкой работы с A64FX. Находящаяся в разработке версия 12 может быть интересна в основном из-за развития в ней реализаций OpenMP 5.0 и OpenACC.

В других публикациях по производительности A64FX иногда анализировалось более широкое число компиляторов, иные версии вышеуказанных компиляторов (и другие опции), в том числе Cray 10.0.1/10.0.2 и ARM 20.3 (т.е. не только 21.0) [39, 57]– [59].

В качестве другого подробного сопоставления компиляторов для A64FX следует указать [52], где широкий набор компиляторов был сопоставлен в работе на 3 разных приложениях на базе классической механики (включая миниприложения, в том числе `minimod` для сейсмического моделирования, реализованного методом конечных разностей [60]; `minimod` применяется как платформа для сопоставления эффективности новых компиляторов в HPC). А использованное для тестов в [52] приложение SWIM (предсказание погоды) входит в состав современной версии SPEC CPU, и используется и для оценок достигаемой производительности на суперЭВМ.

В этой работе тесты проведены на 2 разных моделях A64FX: с частотой 2,2 ГГц — на Fugaku и 1,8 ГГц — на суперЭВМ Oookami (здесь вспомогательных ядер в дополнение к 48 вычислительным в процессорах не было). В этих данных ориентировка для приложений должна быть на общее время выполнения, а не на достигаемое ускорение в зависимости от числа нитей OpenMP, однако причиной пониженной производительности приложения на определенном числе ядер A64FX может быть и недостаточное распараллеливание на OpenMP компилятором. Число использованных нитей OpenMP было кратно 4, плюс тесты с 1 и 2 нитями; общее количество этих нитей равномерно распределялось между CMG.

На Fugaku в тестах использовались более старые, чем 4.5.0, версии компиляторов Fujitsu — а на Oookami компиляторов Fujitsu не было. На Oookami Cray 10.0.1 дал всем приложениям максимальную производительность при любом числе нитей до 48 — кроме minimod, где компилятор Cray дал ошибку. gcc 10.2.0 и 9.3 иногда опережали, а иногда отставали от ARM 20.3; компилятор Cray на Fugaku был недоступен.

FJtrad на Fugaku в мини-приложении *PENNANT*^{URL} (в нем использованы некоторые алгоритмы из давно известного гидродинамического приложения FLAG Лос-Аламосской национальной лаборатории США, что дает шаблоны типичного для FLAG доступа к памяти) генерировал менее производительный код, чем FJllvm, а тот опережал gcc 10.3 при менее 12 используемых нитей OpenMP — здесь ярко выразилось плохое масштабирование производительности Fujitsu-компиляторов версии 4.3.0 с ростом числа нитей. SWIM, оттранслированный Fujitsu 4.4.0a, был всегда быстрее GNU 10.2. А на minimod лучшим оказался FJtrad, затем FJllvm, и медленнее всего — на gcc 8.3.1.

Эти данные, к сожалению, дают меньше сравнительной информации из-за разных используемых версий компиляторов. Вероятно, лучшие производительности обеспечивал Cray 10.0.1, а в качестве вывода в [52] указывается на менее эффективное OpenMP-распараллеливание в компиляторах Fujitsu. К сожалению, в тестируемых в [52] приложениях не применялись SIMD-директивы OpenMP.

В [39] на той же Oookami такие же компиляторы использованы для существенно более широкого круга тестов. Авторы [39] сослались на недостаточный опыт работы с компилятором Fujitsu, и считают, что для C/C++ выбор оптимального компилятора для A64FX затруднен, в том числе из-за возможных ограничений поддержки новых стандартов C++, а для Fortran лучшим выбором посчитали компилятор Cray. В [57, 59] исследована векторизация элементарных функций (квадратный корень, экспонента, синус и другие), и здесь Cray 10.0.1 также был быстрее, чем GNU 10.2 и ARM 20 и 21 (см. рисунок 3, основанный на [7, с. 25]).

Здесь pow означает возведение в степень, recip — расчет обратной величины, а simple, вероятно, относится к примитивному циклу из [59], содержащим оператор

$$y[i] = 2 * x[i] + 3 * x[i] * x[i].$$

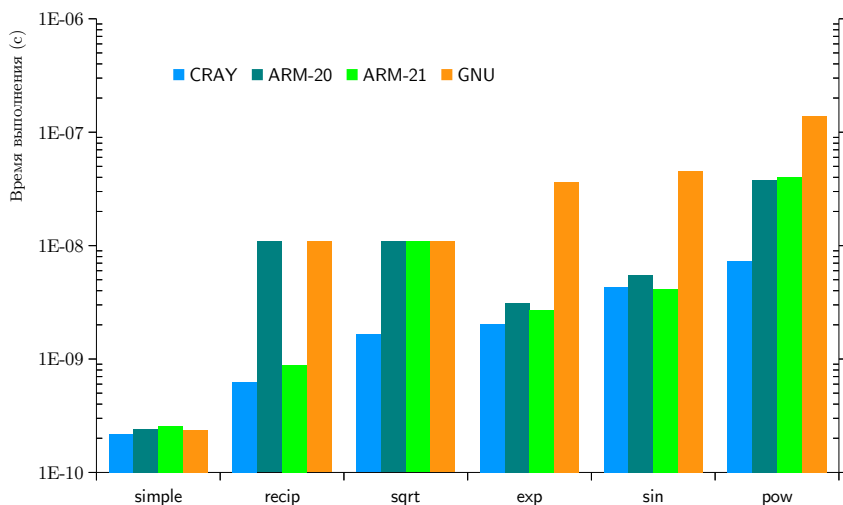


Рисунок 3. Векторизация элементарных операций и функций

Векторизация таких функций позволяет векторизовать и использующие их циклы, которые ранее вообще не векторизовались. Это может использоваться, например, для расчетов базисных функций в квантовой химии. Современные компиляторы Fujitsu, Cray и ARM используют такую векторизацию, а GNU 11.1.0 не векторизовал синус, экспоненту и возведение в степень.

Для векторизации элементарных функций некоторые компиляторы могут использовать библиотеку Sleef (см. ниже) [59]. А лучшие времена выполнения благодаря векторизации таких функций в [59] дал компилятор Fujitsu, существенно опережая Cray. Использувавшийся в [59] GNU 11.1.0 близок к вышеуказанной последней стабильной версии GNU 11.2, и в этой работе предполагается, что такая ситуация с не векторизацией элементарных функций во входящей в GNU библиотеке glibc будет сохраняться еще существенное время.

В [59] на Oookami компилятор Cray немного уступал компилятору Fujitsu в ряде простых, типовых для SVE циклах (существеннее — в векторизованных элементарных функциях), и почти всегда — в тестах *NAS Parallel Benchmarks (NPB)*^{URL} при одноядерном выполнении. А в NPB при распараллеливании на всех 48 ядрах A64FX в одних тестах из NPB компилятор Cray был быстрее Fujitsu, в других — уступал,

но чаще лидером здесь вообще становился GNU 11.1.0. LLVM 11 охарактеризован в [59] как имеющий ограниченную поддержку SVE, а самым слабым местом GNU 11.1 указано отсутствие векторизации ряда элементарных функций. Компилятор Cray чаще уступал GNU в тестах NPВ на 48 ядрах, но опережал GNU в *прокси-приложении для гидродинамики взрывов LULESH 1.0*^{URL}, ориентированном на экса-масштабирование. В качестве вывода в [59] указано на наивысшую достигаемую производительность со средствами от Fujitsu и HPE/Cray.

В [43] в тестах, в том числе SpMV, проведенных на FX1000 с A64FX/2,2 ГГц, сопоставлены fcc 4.4.0a и gcc 10.2.0 — но четкого вывода о преимуществе в производительности, даваемом fcc или gcc, нет. В [41] для задач геномики компилятор FJclang давал большую производительность, чем gcc 11 и ARM HPC — но точные версии компиляторов не указаны.

На основании представленных выше данных о достигаемой при компиляции производительности можно сказать, что сейчас единственным претендующим чаще на роль лидера, дающего максимальную производительность компилятора для A64FX может быть все-таки HPE/Cray, хотя данных для этого совершенно недостаточно. Другой коммерческий компилятор от Fujitsu для такой роли пока не готов, он выигрывает в основном, когда сверхважны реализованные в нем особенности A64FX, а ориентирующийся на HPC LLVM пока не может считаться преимущественным относительно GNU. Да, можно говорить о недостаточной пока зрелости компиляторов для A64FX, но такого четкого лидерства, как у компиляторов Intel для x86-64, для компиляторов A64FX может и вообще не сложиться. И если в тестах производительности с использованием этих компиляторов A64FX опережает серверные процессоры x86-64, то можно утверждать, что и с будущими компиляторами это сохранится.

Для ARM-процессоров доступно большое количество *математических библиотек для HPC*^{URL}. Для A64FX, в том числе для задач линейной алгебры их тоже доступно немало — в том числе Fujitsu SSL2 (Scientific Subroutine Library II), HPE/Cray LibSci, Arm Performance Libraries (ArmPL). По данным [59] к середине 2021 года в ArmPL, Cray LibSci и FFTW, Fujitsu BLAS и FFTW уже применялось SVE. Свободно доступные в исходном тексте библиотеки (например, OpenBLAS) могут быть транслированы с созданием содержащих SVE кодов. Полный список доступных компиляторов и библиотек, а также приложений на Fugaku имеется в [62].

В [39] на нескольких тестах на Oookami сопоставлены в том числе ArmPL, LibSci и OpenBLAS. На DGEMM и HPL в однокитевых вариантах лучшей здесь оказалась ArmPL, LibSci отставала (в DGEMM получено 57,6% и 34,6% от пиковой производительности соответственно), а OpenBLAS — еще сильнее. FFTW в реализации Fujitsu дал 0,93%; LibSci — 0,79%. ArmPL (20.3) не давал SVE-коды, и в FFTW было получено только 0,01% от пиковой производительности. В расчете DGEMM на целом узле LibSci дала 88% от пиковой производительности, а Fujitsu декларировала достижимость 94% [39]. В известных в мире суперЭВМ лабораториях Sandia [63] было показано, что на одном узле Fugaku для DGEMM и ZGEMM библиотека Fujitsu SSL2 дает большую производительность, чем ArmPL, что соответствует данным [59], где в DGEMM ArmPL, LibSci и OpenBLAS отставали по производительности на ядро A64FX от Fujitsu BLAS.

В [45] была исследована производительность DGEMM при работе с маленькими матрицами, уместающимися в кэш L1, при работе с разными библиотеками линейной алгебры. Здесь применялись средства в том числе собственной разработки — loору, и библиотека PSpaMM для умножений разреженных матриц (она может работать и с плотными матрицами), которые генерируют оптимизированные промежуточные коды, а их затем уже можно транслировать например gcc (использовался gcc 11). Но из всех анализировавшихся библиотек SVE использовалось только в loору, и достигнутая одноядерная производительность составила на A64FX/1.8 ГГц всего 32% от пиковой величины. Тем самым [45] демонстрирует высокую важность используемых средств разработки программ.

Для сопоставления можно указать на [57], где для произведения маленьких матриц (неквадратных, произведение матрицы на транспонированную) с такими же процессорами A64FX/1,8 ГГц с использованием библиотеки MADNESS с оптимизацией для малых неквадратных матриц дало 92% от пиковой производительности в таком же однокитевом исполнении. SciLab 10.0.1 давал при этом производительность меньше MADNESS, часто близкую к ArmPL 20.3, хотя иногда и существенно опережая ArmPL.

Возможно, лобовое применение SSL2 в [45] дало бы существенно более высокую производительность. SSL2 включает также и программы для умножения разреженных матриц на вектор (SpMV) с разными форматами хранения разреженных матриц. Однако в [43] найдено,

что тщательная оптимизация SpMV с выбором и формата хранения позволяет существенно превосходить производительность в SSL2 для A64FX.

На A64FX были портированы и другие библиотеки линейной алгебры. Например, в [56] на A64FX применялась библиотека MAGMA, известная своей ориентацией на работу с GPU, а в [64] — BLIS [65]. Последняя библиотека, отличающаяся расширенным набором функций для операций с плотными матрицами по сравнению с BLAS, доступна в исходных текстах и сооружает оптимальные для определенного набора архитектур, включая A64FX, модули. *По экспериментальным данным*^{URL}, BLIS в однонитевом и многонитевых вариантах обычно опережает SSL2, ArmPL и Eigen, хотя иногда уступает SSL2.

Библиотека EigenExa для решения задачи на собственные значения со сведением матрицы к трехдиагональной методом Хаусхолдера, и последующим применением метода «разделяй и властвуй» (Divide-and-Conquer, DC) [66] ранее работала еще на К-компьютере, а теперь перенесена на Fugaku с модернизацией DC-части, базировавшейся на DC от ELPa, и ее масштабирование при MPI-распараллеливании проверялось до 32768 узлов [66]. Среди прочего можно отметить и библиотеку SLEEF, где векторизация используется для расчетов элементарных функций; SVE поддерживается там с 2018 года [67].

Но вообще влияние библиотек на получаемую в тесте производительность не следует и переоценивать. Так в HPL, где работа с DGEMM лимитирует производительность, применение LLVM дало на 5% больше производительность, чем транслированные компиляторами Fujitsu коды, хотя во всех этих расчетах использовалась SSL2 [50].

Другими важнейшими для работы с A64FX являются средства распараллеливания. Понятно, что разные реализации OpenMP доступны для A64FX просто вследствие доступности ряда разных компиляторов. Анализ уровня достигаемого в OpenMP распараллеливания проведен в [39, 52]. Как было отмечено выше, для создания приложений естественнее ориентироваться на достигаемую производительность, а не собственно на уровень распараллеливания по OpenMP.

На A64FX можно работать с целым набором разных MPI-реализаций, в том числе Fujitsu MPI (на базе OpenMPI), RIKEN-MPICH, HPE/Cray MPI (базируется на MPICH) и естественно с MVAPICH2 и OpenMPI. Как справедливо отмечено в [68], хорошее тестирование должно включать сочетание компилятора с реализацией MPI. В [68]

на Oookami на известном в мире суперЭВМ приложении FLASH (с применением гидродинамики для астрофизики) лучшим по времени выполнения оказался Cray 10.0.1 с поддержкой SVE; gcc 10 ему уступал, Fujitsu еще, а ARM 21 отстал уже в несколько раз. Варианты Cray и gcc работали при этом со транслированными соответствующими компиляторами MVARICH 2.3.5 и OpenMPI 4.0.5; последняя была чуть медленнее MVARICH. В [59] предположено о наличии недостатков Fujitsu MPI при работе не с TofuD, а с Infiniband HDR, что выразилось в более плохом масштабировании (с числом узлов) в HPL и FFTW на их Fujitsu-реализации по сравнению с ArmPL.

В [39] на Oookami с межсоединением Infiniband HDR200 успешно протестированы различные комбинации компиляторов и MPI-реализаций (MVARICH 2.3.4 и OpenMPI 4.0.5). В тестах OSU MPI между двумя узлами (точка-точка) с применением OpenMPI достигнутая двунаправленная пропускная способность составила 19,4 ГБ/с (около 78% от пиковой), однонаправленная — 12,3 ГБ/с (передавалось по 8 МБ); задержки PUT/GET (передачи по 8 байт) составили 3,9/5,2 мкс. В [56] MPI-коммуникации точка-точка на суперЭВМ Summit (второе место в TOP500 после Fugaku) при разных размерах сообщений давали более низкую задержку, чем на Fugaku.

Поскольку A64FX изначально планировался на применение в суперЭВМ, здесь уже можно использовать и популярные PGAS-средства распараллеливания OpenSHMEM. Полученные в [39] задержки PUT/GET составили тут 4,5/4,1 мкс. К сожалению, нет данных о возможности использовании на A64FX при работе с Infiniband *новых средств Nvidia/Mellanox HPC-X^{URL}*, в которых кроме поддержки MPI, OpenSHMEM имеется еще и другие оригинальные разработки. В [69] именно при работе с HPC-X была получена наивысшая производительность в ряде тестов на HDR200. Здесь в двухпроцессорных серверах также, как в A64FX, применялись PCIe-v3 x16, но с одной общей платой HDR на сервер. Достигнутые в тестах OSU для всех использованных реализаций MPI (т.ч. и для MVARICH2-2.3.1) задержки GET, а на MVARICH2 — односторонние пропускные способности GET/PUT, а также двусторонние пропускные способности PUT при работе с серверами на разных процессорах Xeon Haswell и Cascade Lake здесь были существенно лучше, чем при работе на серверах с одним A64FX в [39].

Для сложных HPC-задач, которые могут выполняться на различных вычислительных системах и используют распараллеливание, возникает и проблема переноса из одной аппаратной среды в другую. Для решения этой проблемы применяются использующие C++ средства Kokkos, создающие свой высокий уровень абстракций для иерархии вычислений и памяти [70]. Kokkos планируется в качестве возможного применения в EFLOPS-суперЭВМ; естественно, что эти средства работают и с A64FX. Начиная с Kokkos версии 3.3, там может применяться компилятор Fujitsu; в SIMD-библиотеку Kokkos была добавлена поддержка SVE. Однако в [63] найдено, что хотя работа с SVE ускоряется относительно применения NEON, но Kokkos работает быстрее на Xeon Skylake, чем на A64FX.

3. Производительность A64FX для HPC-задач

3.1. Производительность A64FX на уровне большого числа узлов суперЭВМ

В начале — краткая информация о производительности на уровне всей суперЭВМ Fugaku, что позволяет продемонстрировать достижимый уровень масштабирования производительности, и возможности построения/применения суперЭВМ без акселераторов. Приводимые далее данные взяты с соответствующих сайтов по TOP500/GREEN500 и GRAPH500 (июньские версии 2021 года). Исходные показатели производительности Fugaku в TOP500 [13] в 2020 году были несколько увеличены, что, очевидно, является следствием прогресса в программном обеспечении. Fugaku имеет 158976 узлов, из которых 152064 участвовали в расчетах для тестов производительности в TOP500. При пиковой производительности 537 PFLOPS (A64FX в Fugaku работают на частоте 2,2 ГГц) в HPL получено 442 PFLOPS (это 86% от пиковой производительности всех узлов, участвовавших в расчетах). В тесте HPL-AI, где используется смешанная точность (в том числе половинная), был достигнут уровень 2,0 EFLOPS. Находящаяся на втором месте по обоим этим тестам суперЭВМ Summit (с Power9 и V100 в узлах) имеет 149 PFLOPS в HPL и 1,15 EFLOPS в HPL-AI (последний результат был получен только в 2021 году). В HPCG Fugaku опережает занимающую второе место Summit гораздо больше — 16 против 3 PFLOPS, что составляет 2,8% и 1,3% от пиковой производительности соответственно [13].

Ранее на Fugaku с работой на A64FX/2,0 ГГц в тесте HPL-AI была достигнута производительность 1,4 EFLOPS, что стало первым достижением EFLOPS-уровня, хотя и в расчете с пониженной точностью [71, 72]. И это все равно выше, чем полученное в Summit.

По энергоэффективности в HPL Summit чуть-чуть отстает от Fugaku: 14,72 против 14,78 GFLOPS/Вт у Fugaku, но в TOP500 теперь есть суперЭВМ с гораздо более высокими показателями. GREEN500 сейчас возглавляет также японская суперЭВМ MN-3, содержащая японские акселераторы в серверах с Xeon 8260M, однако в TOP500 она на 336-м месте. Со второго по 9-е места в GREEN500 занимают суперЭВМ с серверами на AMD EPYC Rome или Milan разных моделей с GPU NVIDIA A100. Из них отметим 6-е место в GREEN500 суперЭВМ Perlmutter, которая занимает еще и пятое место в TOP500. Fugaku, лидировавшая в GREEN500 в конце 2019 года, сейчас находится на 20-м месте. Это демонстрирует не только успешность AMD с продвижением A100 в последнее время, но и существенно более частые усовершенствования серверных процессоров x86-64 по сравнению с процессором Fujitsu. Но его возможная модернизация может отличаться более сильным ростом производительности (см. про его потенциальную модернизацию в Заключение).

GRAPH500 [73] демонстрирует производительность при обработке больших массивов данных, и реализует задачу поиска в графе. Вычислительные системы RIKEN лидируют в GRAPH500 с 2014 г., но ранее это был K-компьютер, а теперь в BFS-поиске [73] лидером здесь является Fugaku. По аналогии с GREEN500 в GRAPH500 теперь измеряется энергоэффективность; Fugaku здесь не представлен, а в первом десятке — две вычислительные системы из России. Подробнее работа с GRAPH500 при использовании BFS-ядра на Fugaku рассмотрена в [74], а про оптимизацию при этом энергопотребления см. в разделе 3.2.

Приведем также данные о производительности на FUGAKU в тесте, который актуален для задач квантовой хромодинамики (КХД). КХД-расчеты выполнялись с применением смешанной точности. На 147456 узлах была достигнута производительность 102 PFLOPS (около 10% от пиковой SP-производительности) при потребляемой мощности 20 МВт [75, 76]. Дополнительная информация об этих тестовых расчетах приведена в разделе 3.3.

Укажем также на первоначальные данные [56] о сравнении производительности суперЭВМ Summit и Fugaku на известном CFD-приложении *NEK5000*^{URL}, входящим в набор программных средств *CEED*, ориентированный на экза-масштабирование^{URL}. Summit дал здесь лучшие показатели, а у Fugaku на больших задачах возникли проблемы ввода-вывода. Масштабируемость производительности на Fugaku с A64FX/2,0 ГГц при количестве узлов до 2048 в других приложениях проиллюстрирована в [38, с. 19], см. рисунок 4. Информация о приложениях на A64FX и данные об их производительности приводится ниже в разделе 3.3.

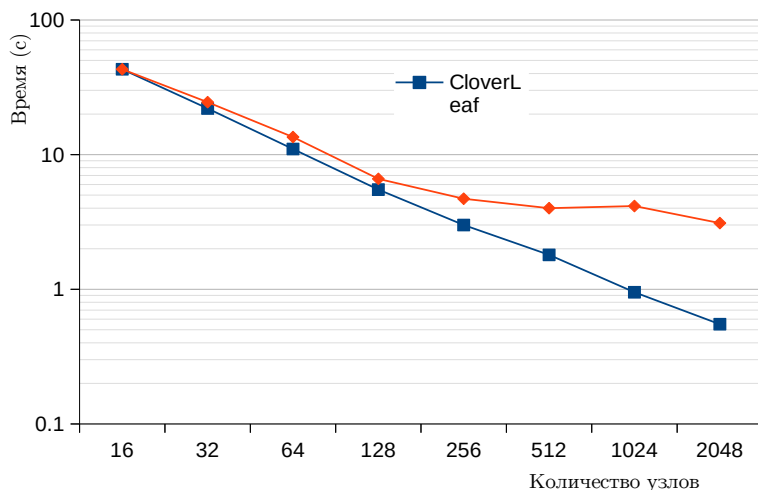


РИСУНОК 4. Производительность CloverLeaf и TeaLeaf

Здесь следует указать также на расчеты по модели атмосферы NICAM, проведенные с использованием 131072 узлов Fugaku (82% от всех узлов), где с применением одинарной точности была достигнута производительность в 79 PFLOPS [77].

В качестве другой классической задачи, которая распараллелена между соединенными TofuD узлами Fugaku, следует указать и задачу на собственные значения для плотных матриц, которая может применяться в разных НРС-областях, в том числе в CFD и квантовой химии. В квантовой химии часто считается нужной диагонализация матриц больших размерностей (N), например, для больших молекул, хотя в некоторых современных методах квантовой химии,

в том числе в наиболее массовом DFT, для больших молекулярных систем от требующей $O(N^3)$ диагонализации отказались, разработав приближенные линейно масштабируемые методы. Оба используемых в тестировании для диагонализации с помощью библиотеки EigenExa этапа — сведения к трехдиагональной матрице и последующей ее диагонализации методом DC требуют существенных вычислительных ресурсов. Однако метод Хаусхолдера плохо распараллеливается и требует большой пропускной способности памяти — а DC хорошо распараллеливается, но часто дает разбалансировку нагрузки разных параллельных процессов [66]. Для матриц с $N > 250000$ при распараллеливании OpenMP+MPI было получено приемлемое масштабирование производительности с использованием до 16384 узлов [66].

3.2. Данные тестов производительности A64FX в вычислительных системах, содержащих не более нескольких узлов

Далее речь идет только о производительности от одного ядра A64FX до нескольких вычислительных узлов — хорошая возможность для масштабирования на уровне большой суперЭВМ Fugaku практически очевидна. В качестве стартовых величин для оценки данных тестов производительности в расчете на узел (они часто производились на Fugaku и Oookami) укажем приведенные в [13] оценки для stream triad — порядка 830–840 ГБ/с, а для DGEMM 2,4 TFLOPS (Oookami, Cray SciLab) и более 2,5 TFLOPS (на Fugaku).

Приводимые ниже данные по производительности A64FX содержат в основном только максимально полученные в публикациях значения; возможный отбор оптимальных алгоритмов, компиляторов/опций/библиотек и так далее здесь не рассматривается. Данные о производительности здесь относятся к диапазону от одного ядра до нескольких серверов в кластере (и соответственно нескольких процессоров A64FX). Когда имеются соответствующие данные, производится сравнение производительности с процессорами x86-64: Xeon Skylake и Cascade Lake; AMD EPYC Zen 2 и 3, а также при работе на GPU V100 и A100. Сопоставление производительности с более старыми процессорами или ускорителями не рассматривается как гораздо менее актуальное (а данных относительно MI100 пока нет). Иногда приводятся сопоставительные данные с первым используемым в суперЭВМ ARM-процессором ThunderX2, однако преимущества производительности A64FX относительно него в среднем очевидны.

Рассмотрение данных тестов производительности A64FX начинается с тестов DGEMM (и других умножений плотных матриц), поскольку это во многом определяет производительность в используемом в TOP500 тесте HPL. Но DGEMM актуальна для самых разных приложений, например, в вычислительной химии — в молекулярной динамике или в разных неэмпирических методах квантовой химии с учетом электронной корреляции. По данным [34], на A64FX/1,8 ГГц в DGEMM можно получить больше 2,5 TFLOPS (90% от пиковой производительности). При этом уменьшение частоты с 2,0 до 1,6 ГГц уменьшает производительность на 20%, а энергопотребление — на 18%; включение эко-режима уменьшает производительность уже на 50%, а энергопотребления — на 16% [34]. В [38] для A64FX/2,2 ГГц указано на производительность 3,2 TFLOPS на 200 Вт. На рисунке 5 согласно [38, с.23] представлены графики достигаемой производительности (в TFLOPS) и энергопотребления (Вт) в зависимости от числа используемых нитей в DGEMM. Полученные в [5] данные об

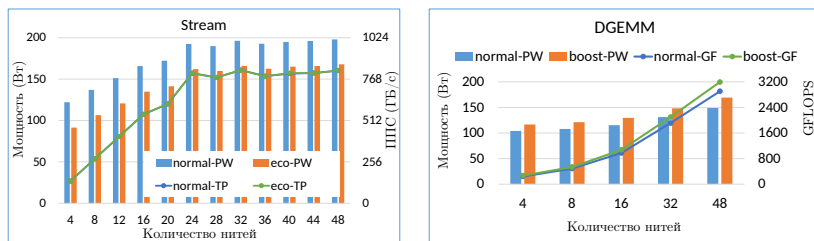


РИСУНОК 5. Энергопотребление и производительность в stream и DGEMM

уровнях энергопотребления компонентов A64FX при работе с DGEMM показывают, что основная доля энергопотребления приходится на процессорные ядра, кэш L2 потребляет в разы меньше, работа с HBM2 — совсем мало.

Поскольку A64FX ориентирован на работу с HPC без подсоединения акселераторов, полезно сопоставить производительность DGEMM на A64FX и на GPU V100 и A100. По данным [4] на V100 при этом достигается 7,2 TFLOPS, а в A100 производительность явно больше; подробнее информация по тестам производительности A100 представлена в [47].

Максимальная производительность DGEMM на ядро A64FX достигается с применением SSL2, соответственно данные для сопоставления с x86-64 процессорами приводятся здесь для A64FX с SSL2. В A64FX/1,8 ГГц было получено 40,9 GFLOPS (71% от пиковой производительности) — что меньше, чем у ядра Xeon Skylake 8160 (43,4 GFLOPS и 97%), но выше, чем у EYUC 7742 (25,3 GFLOPS и 70%) [59], в котором поддерживаются только вектора по 256 бит. В [39] расчет на таком же узле с A64FX дал пониже производительность, вероятно из-за неиспользования SSL2.

Имеются также данные о производительности A64FX в других произведениях матриц, отличных от традиционной (для HPC) DGEMM. Так, в [63] приводятся данные о производительности ZGEMM на A64FX из Fugaku. Были проведены и специальные исследования производительности на A64FX умножений маленьких матриц, в том числе умещающихся в кэш L2 в A64FX, с использованием и разных библиотек линейной алгебры [45]. В [45] не были достигнуты достаточно близкие к пиковой производительности A64FX/1,8 ГГц результаты — только 1,7 TFLOPS, но более новые пакетные варианты BLAS/DGEMM [78], ориентированные на работу с маленькими матрицами, или хорошо оптимизированная библиотека SSL2 здесь не использованы. Данные о производительности A64FX/1,8 ГГц для умножения небольших матриц на транспонированную с использованием специализированного алгоритма получены в [57]. Он для одного ядра дал 53,2 GFLOPS (около 92% от пиковой производительности), опережая ядро Xeon Skylake/3,7 ГГц с MKL. А в [53] исследована производительность специального для задач квантовой хромодинамики умножения матриц, где DGEMM не используется.

Данные теста HPL на уровне Fugaku (в том числе в современных вариантах из TOP500) приведены выше в разделе в разделе 1. Для Oookami с A64FX/1,8 ГГц производительность HPL на узел составила 1129 GFLOPS (40,8% от пиковой величины), уступая двухпроцессорному серверу с Xeon 8160 (1156 GFLOPS и 53,7% соответственно), содержащим те же 48 ядер, как у A64FX. А двухпроцессорный сервер с EYUC 7742 со 128 ядрами по производительности существенно обошел их обоих (1911 GFLOPS и 41,5% соответственно). С ростом числа узлов до 4–8 преимущества в производительности кластеров на базе x86-64 сохранялись [59]. А по энергоэффективности в HPL в конце 2019 года суперЭВМ на базе прототипов A64FX/2,0 ГГц при не очень

больших размерностях заняла первое место в TOP500, обогнав и системы с акселераторами [79]. В [39] производительность в HPL на таком же узле с A64FX была ниже, но SSL2 не использовался.

В вышеприведенных тестах A64FX по производительности нередко отставал от современных процессоров x86-64, и следует теперь обратить внимание на тесты, дающие данные о производительности памяти (ее пропускной способности), где у A64FX должны быть четкие преимущества. Практически стандартом для этого можно считать *stream*^{URL}, включающий 4 теста в виде циклов для работы с одномерными массивами — их копирование (copy), умножение на скаляр (масштабирование, scale), суммирование двух массивов в третий (add) и наиболее часто используемый для тестов triad:

$$A[i] = B[i] + s * C[i]$$

где s — скаляр. У этого теста есть ряд модернизированных вариантов, в том числе BabelStream [80], ориентированный в первую очередь на работу с GPU (при этом время передач, например, по PCIe, не учитывается) или на устройства с весьма большим числом ядер, в котором в том числе добавлен еще тест для скалярного произведения векторов. В набор микротестов lmbench [81] входит и stream с исходными данными из этого набора. В случае, если достигнута пропускная способность в stream для A64FX и в альтернативных аппаратных средствах мало зависит от конкретного варианта stream, уточнение этого варианта здесь в тексте может быть опущено. Но про использование DAXPY, где результат в приведенном выше операторе цикла stream triad записывается в тот же массив B, а не в другой массив A, указывается однозначно.

Указанные выше стартовые оценки [13] для stream triad в A64FX на Fugaku близки к данным тестов BabelStream для V100 800-840 ГБ/с, а на A100 пропускная способность в BabelStream гораздо больше: 1,3–1,4 ТБ/с [47]. Пропускная способность у такого процессора A64FX (около 840 ГБ/с) в [43] была гораздо больше, чем у двухпроцессорного 96-ядерного сервера с Cascade Lake (Xeon 9242), где было получено 420 ГБ/с, и близка к пропускной способности V100. В [43] изучена и пропускная способность stream triad в зависимости от числа используемых ядер внутри одной CMG.

В [79] на узле Fugaku в stream triad было получено около 80% от пиковой пропускной способности. В [34] для сервера из Fugaku

получена близкая к данным [43] пропускная способность в stream triad, и показано, что включение эконо-режима практически не уменьшает пропускную способность, но уменьшает энергопотребление на 18%.

В [41] пропускная способность в узле Fugaku на lmbench/stream сопоставлена с двухпроцессорным сервером с Xeon 8160 с общим числом процессорных ядер, как у A64FX, при этом исследовалось и влияние локализации памяти. A64FX опережал сервер со Skylake где-то в два и более раз. В [82] сопоставлена пропускная способность stream copy у A64FX/2,2 ГГц и ThunderX2 в зависимости от числа ядер; результаты A64FX были в разы больше. В [5] было найдено, что в тестах stream на A64FX основное энергопотребление приходилось на процессорные ядра; HBM2 потребляла заметно меньше, а кэш L2 — в разы меньше.

В [39] stream triad дал 830 ГБ/с на Oookami для A64FX/1,8 ГГц, а в DAXPY удалось получить 840 ГБ/с. Это было в разы быстрее двухпроцессорного сервера с Xeon 8160. На одном ядре A64FX DAXPY дал 53 ГБ/с — в 2,5 больше, чем на сервере с Xeon 8160. В [53, 54] для A64FX/2,2 и 1,8 ГГц получены данные для всех четырех тестов stream в зависимости от числа задействованных ядер A64FX; максимально достигнутая пропускная способность составила 835 ГБ/с, а в [43] максимум для узла Fugaku был 841 ГБ/с.

В [57] исследована достигаемая A64FX в Oookami пропускная способность памяти в DAXPY как в однопитевом, так и в многопитевом варианте (с распараллеливанием в OpenMP), см. ниже (таблицу 5). Уже однопитевой результат в A64FX гораздо больше, чем в ядре Xeon Skylake/3 ГГц. Найдено, что около 6 потоков на CMG могут насыщать пропускную способность. В общем, все данные тестов пропускной

ТАБЛИЦА 5. Сопоставление пропускной способности (ПС, ГБ/с) в DAXPY: A64FX/1,8 ГГц и Xeon Gold 6136/3 ГГц — 24 ядра SP (Skylake)

Число ядер	A64FX		Skylake	
	ПС	байт/FLOP	ПС	байт/FLOP
1 ядро	53,0	0,92	21,1	0,17
1 сокет	840	0,30	145	~ 0,06

способности памяти в A64FX подтверждают большое аппаратное преимущество HBM2 относительно памяти DDR4 в процессорах

Хеон и ThunderX2. Но есть еще специальный, распараллеленный в OpenMP+MPI, тест для пропускной способности памяти, где результат во многом зависит от задержки памяти — *mega-sweep*^[URL]. И здесь A64FX проигрывает по такой суммарной пропускной способности двухпроцессорному (вероятно, с Хеон 6126) серверу с DDR4 (SKL на рисунке 6), имеющему в 2 раза меньшее число ядер — из-за более высоких задержек памяти HBM2 [83]. Но двухпроцессорный сервер с ThunderX2 (56 ядер/2,6 ГГц — TX2 на рисунке 6), имеющий такую же память DDR4, от A64FX существенно отстает — поскольку у него вектора в 2 раза короче, чем в A64FX, а векторизация в этом тесте важна.

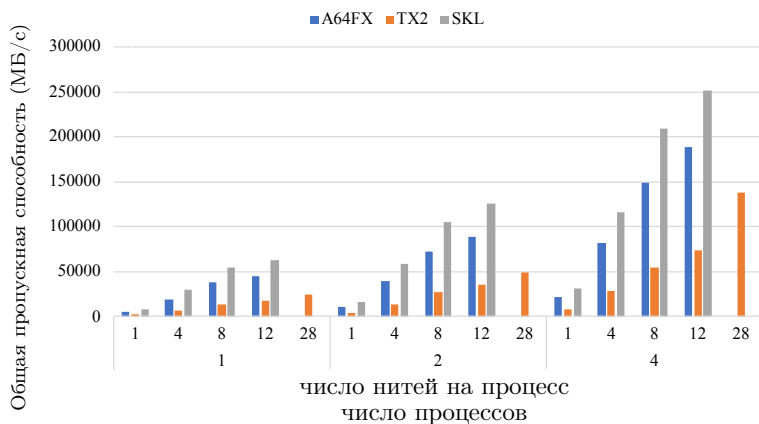


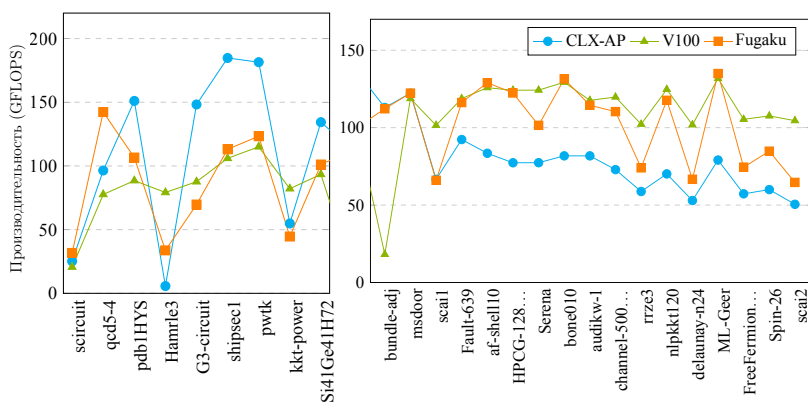
Рисунок 6. Общая пропускная способность в mega-sweep (по [91, рис.7])

Умножение матрицы на вектор также бывает актуальным для разных приложений, например, CFD. Для умножения плотной матрицы на вектор ограничивающей производительность может быть пропускная способность памяти, поэтому A64FX может получить преимущество над x86-64. В [84], где для работы приложения это было актуально, был проведен и анализ производительности BLAS-программы SGEMV, и найдено, что A64FX/2,2 ГГц (из FX1000) в несколько раз отстает от GPU A100 и от японской вычислительной системы NEC TSUBASA на базе процессоров SX-Aurora, поддерживающих работу с длинными векторами [85]. Однако A64FX существенно опередил при этом двухпроцессорные серверы с Хеон 6248/2,5 ГГц (всего 40 ядер) и с AMD

ЕРУС Rome 7702/2,2 ГГц (всего 128 ядер). В [43] на A64FX в FX1000 была исследована одна из конкретных реализаций умножения плотной матрицы на вектор, и найдено, что применение секторного кэша позволяет задавить деградацию производительности при увеличении длины вектора.

Умножение разреженной матрицы на вектор (SpMV) может еще более актуально для разных приложений, например, для квантовой химии (в методе конфигурационного взаимодействия) или квантовой хромодинамики. Возможно, решение задачи умножения разреженной матрицы на вектор на A64FX впервые было предпринято еще с применением эмуляции процессора A64FX, в [86]; однако здесь рассматриваются только расчеты на реальном процессоре. В качестве наибольшей достигнутой производительности в SpMV можно, наверное, указать на порядка 220 GFLOPS, что было получено на A100 [47].

На рисунке 7 по [87, рис. 16], см. также [43], показана достигаемая производительность в SpMV для больших и маленьких матриц с различными указанными на горизонтальной оси выборами исходных данных — для A64FX/2,2 ГГц, V100 и двухпроцессорного сервера с Xeon 9242 (всего 96 ядер; на рисунке — CLX-AP). В сопоставительных расчетах



(a) малые матрицы

(b) большие матрицы

РИСУНОК 7. Производительность SpMV на A64FX, V100 и Xeon 9242 (CLX-AP)

применялся формат SELL-C- σ , в котором на A64FX достигалась наивысшая производительность [43]. Производительности V100 и

A64FX оказались достаточно близкими (заведомо меньше 150 GFLOPS), а CLX-AP нередко лидировал по производительности на маленьких матрицах, но уступал на больших матрицах. Границей между большими и маленькими матрицами была выбрана суммарная емкость кэшей L2/L3 в CLX-AP.

В [43, 87] проведен также анализ режимов энергопотребления на производительность A64FX в SpMV. Понижение частоты с 2,2 до 2,0 ГГц понижает производительность в среднем на 2,7%, но энергопотребление понижается примерно на 13%. Включение еще и эко-режима понижает энергопотребление еще на 21% (всего — на 31%).

В [88] изучалась производительность SpMV на A64FX/1,8 ГГц, достигаемая в FX700. При масштабировании до 12 ядер (в CMG) было достигнуто 31 GFLOPS, а на всем узле максимальная производительность была 131 GFLOPS.

HPCG является другим тестом для HPC, где производительность существенно зависит и от пропускной способности памяти [89]. В

ТАБЛИЦА 6. Сравнение производительности вычислительных систем на серверных процессорах ARM и x86-64 в тестах HPCG и OpenSBLI

Система	1 узел	2 узла	4 узла	8 узлов	% от пиковой производительности
Тест HPCG (GFLOPS)					
A64FX/2,2 ГГц ¹	38,3	78,9	157,5	313,5	1,1
EPCC NGIO ²	37,6	73,9	147,9	292,6	2,0
Fullhame ³	33,8	67,7	133,3	261,3	3,0
Тест OpenSBLI (время расчета в секундах)					
A64FX/2,2 ГГц	3,4	1,9	1,0	0,7	
EPCC NGIO	1,2	0,8	0,5	0,3	
Fullhame	1,2	0,7	0,6	0,3	

¹с TofuD; ²с FDR Infiniband; 2× Xeon 8260M; ³с Aries; 2× ThunderX2/2,2 ГГц

таблице 6 представлены данные из [89] о производительности в Fugaku — до 8 узлов с A64FX/2,2 ГГц, связанных через TofuD, по сравнению с кластерными системами из двухпроцессорных серверов: на базе Cascade Lake (связанными через Intel OmniPath) и на базе 32-ядерных ThunderX2/2,2 ГГц (связанными через Infiniband EDR). При этом для Xeon и ThunderX2 была проведена оптимизация исходных кодов HPCG,

дававшая существенный прирост производительности — а на A64FX этого не делалось.

Поэтому демонстрируемые в этой таблице преимущества A64FX в производительности HPCG относительно ThunderX2 и Cascade Lake могут быть еще больше. При этом число ядер в узле с A64FX совпадает с числом ядер в узле с Cascade Lake, и меньше, чем в узле с ThunderX2. У A64FX опережение Xeon по производительности становится более выраженным на нескольких узлах, что может быть связано и с эффективной работой с TofuD [89].

Имеется два набора циклов, созданных в Ливерморской лаборатории США, которые используются в тестах для HPC. Классический давно используемый набор [90] на Fortran содержит типовые циклы из некоторых программ для HPC, применяемые в первую очередь для оценок производительности при векторизации. Этот набор был использован в [34] для исследования производительности на одном ядре A64FX/2.2 ГГц.

Другой набор, *RAJA Performance Suite*^{[URL](#)}, включающий ядра с циклами на C++ (в том числе и скалярное произведение векторов) для HPC с распараллеливанием в OpenMP, был использован в [83] для сравнения производительности A64FX/2,0 ГГц из FX1000 с двухпроцессорными серверами с ThunderX2/2,0 ГГц (всего 56 ядер) и Xeon Skylake/2,6 ГГц (всего 24 ядра — вероятно, Xeon 6126) при разном числе нитей. В трех из четырех использованных из RAJA ядрах A64FX в разы опережал Skylake, а ThunderX2 от A64FX всегда сильно отставал.

Вообще используемые в HPC тесты производительности иногда включают в себя отдельные компоненты, которые сами являются другими известными тестами. Например, HPCC [91] включает HPL, DGEMM, stream и другие тесты. В [39, 59] рассматриваются данные о достигаемой производительности на нескольких тестах из HPCC, в том числе DGEMM и HPL (они были рассмотрены выше), а также в БПФ (в FFTW). Наибольшая производительность на узле с A64FX/1,8 ГГц (26 GFLOPS — 0,9% пиковой производительности) была достигнута при использовании FFTW3 от Fujitsu [39]; в [59] — чуть ниже. При этом двухпроцессорные узлы с x86-64 процессорами показали более высокую производительность: 46 GFLOPS (1,5% от пиковой) на Xeon 8160 — также с 48 ядрами; 72 GFLOPS (1,6% от пиковой) на EPYC 7742 — со 128 ядрами на сервер. При распараллеливании на несколько узлов (до

8) четкие преимущества процессоров x86-64 по производительности сохранялись [59].

Еще одним знаменитым тестом производительности, который выполнялся и на A64FX, является NPВ, который также включает в себя ряд вычислительных компонент, большинство которых основано на кодах из CFD. У NPВ имеется и вариант с OpenMP-распараллеливанием [92], который был использован в тестах A64FX/1,8 ГГц в [59], где тестирование включало 6 компонент из NPВ. На одном ядре A64FX сильно проиграл по производительности ядру Xeon 8160; при использовании всех ядер A64FX превзошел Skylake на двух компонентах из NPВ, а в других компонентах NPВ его отставание уменьшилось. Уровень распараллеливания на A64FX был выше, чем на Skylake. A64FX лучше выглядел на компонентах NPВ, где важна пропускная способность памяти, а Skylake выигрывал в более вычислительно интенсивных расчетах.

Наиболее массовыми тестами производительности следует считать, наверное, SPEC сru2017. В [93] для A64FX измерялась производительность в SPECrate, однако это базировалось на использовании симулятора A64FX на базе gem5, и эти результаты здесь не рассматриваются. В [50] были получены данные для A64FX SPECspeed — как целочисленного, так и с плавающей запятой, и SPEC OMP2012, но приведенные в [50] величины являются относительными и демонстрируют лишь условную производительность, полученную одним компилятором по отношению к другому. В [94] найдено, что A64FX в Fugaku в SPEC sru2017 отставал по производительности от двухпроцессорного сервера с Xeon 8168 (всего 48 ядер на сервер), а в тестах SPEC OMP отстал и от 28-ядерного Xeon (причиной этого указана поддержка SMT-режима в Xeon). Но в некоторых тестах SPEC sru2017 с плавающей запятой, и SPEC OMP из-за более высокой пропускной способности памяти A64FX имел более высокую производительность, чем Xeon.

Про основные результаты тестов GRAPH500 на Fugaku уже сказано в разделе в разделе 3. Эти тесты подробно рассмотрены в [74, 95, 96], где исследована зависимость производительности BFS от числа узлов, и указано на особенности использования топологии TofuD. Исследована также зависимость производительности от выбора режима энергопотребления. Поскольку BFS не работает с числами с плавающей запятой, включение эконо-режима ожидалось эффективным решением. Оптимальное сочетание высокой производительности и

уровня энергопотребления дал соответственно убыстренный режим + эко-режим.

Еще одним проведенным на A64FX из Fugaku тестом является *HIMENO^{URU}*, он ориентирован на CFD (анализ несжимаемой жидкости), и там решается уравнение Пуассона итерационным методом Якоби. Этот тест стал достаточно распространенным в HPC; он выполнялся и на SX-Aurora TSUBASA, имеется и вариант для кластера с GPU в узлах. По данным [13], на A64FX было получено 346 GFLOPS, на V100 — 305 GFLOPS, на SX-Aurora TSUBASA 286 GFLOPS, а на двухпроцессорном сервере с Xeon 8168 — всего 85 GFLOPS.

В [97] на A64FX/1,8 ГГц была исследована производительность на усовершенствованном алгоритме сортировки с использованием векторизации с работой на больших и небольших массивах. Достигнутое ускорение за счет применения этого алгоритма было ниже, чем полученное на Xeon 8170 с AVX-512.

Выше в разделе про средства разработки программ на A64FX было указано на некоторые недоработки в компиляторах для A64FX в векторизации элементарных функций. В [59, рис. 2] имеются сравнительные данные о производительности A64FX/1,8 ГГц и Xeon 6140 (см. рисунок 8). На этом рисунке используются те же обо-

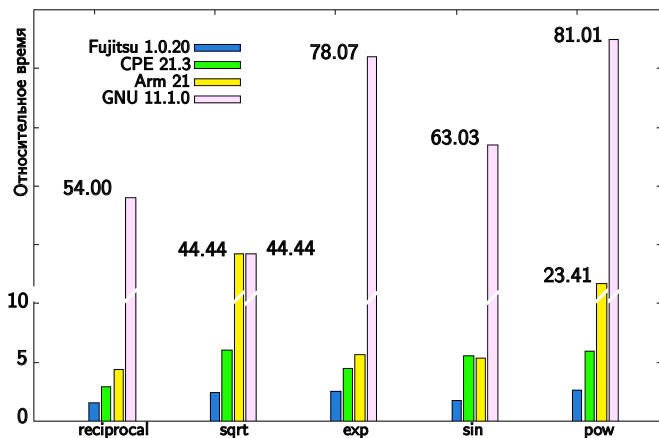


РИСУНОК 8. Время выполнения на A64FX (по отношению к Skylake) элементарных функций

значения, что и на аналогичном рисунке 3, а CPE означает среднюю

программирования C_{ray}. Видно, что здесь Skylake имеет преимущества в производительности.

В целом приведенные данные тестов показывают, что A64FX нередко уступают по производительности процессорам x86-64, а более новые Xeon Ice Lake могут в этом плане быть еще лучше. Преимущества A64FX в производительности для HPC чаще относятся к случаям, когда производительность ограничивается пропускной способностью памяти.

3.3. Данные о производительности A64FX на приложениях

В начале этого раздела проиллюстрируем доступный в ряде публикаций рисунок, показывающий повышенную относительно Xeon Cascade Lake с тем же числом ядер (относительно двухпроцессорного сервера с Xeon 8268/24 ядра 2,9 ГГц) производительность A64FX/2,2 ГГц при почти в два раза более низком энергопотреблении [38, с. 24] (рисунок 9). По производительности A64FX во всех приложениях

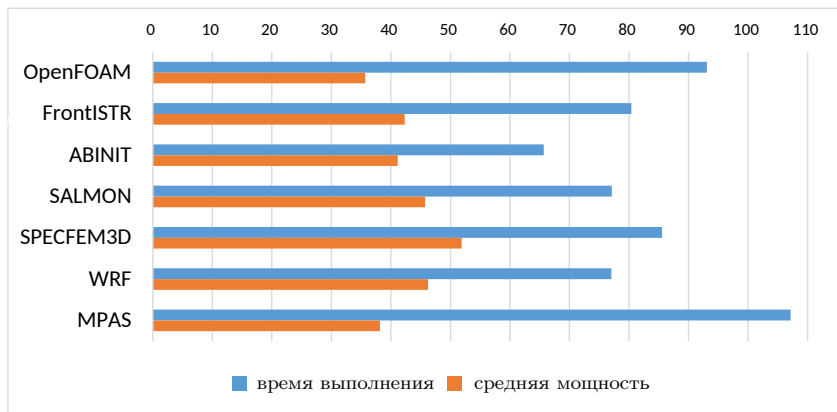


Рисунок 9. Данные относительной производительности A64FX в приложениях

(кроме приложения предсказания погоды, MPAS) опережал Cascade Lake, хотя у Intel есть и более мощные процессоры, в том числе более новое семейство Xeon Ice Lake, и надо сопоставлять и стоимостные показатели.

Данные, представленные на этом рисунке, относятся к разным приложениям, доступным в исходном тексте. Для понимания того, что дает этот рисунок, важно то, что именно считалось — от этого может зависеть, что является лимитирующим время расчета, а в приложениях квантовой химии вообще могут использоваться кардинально отличающиеся и в плане производительности методы. Этот рисунок демонстрирует разные расчеты по разным приложениям, в которых A64FX может быть эффективнее, чем Cascade Lake, например, из-за более высокой пропускной способности памяти. Для HPC-приложений может быть нужна более детальная информация, а краткую дополнительную информацию по тестам, представленным на данном рисунке, можно найти в [34].

Кроме данных из [38], относящихся к A64FX из FX1000, сопоставление с тем же сервером с Cascade Lake имеется для A64FX из FX700 [98], где оно включает 4 приложения из 7, использованных в [38]. На двух приложениях по вычислительной химии (LAMMPS — для молекулярной динамики, и Quantum Espresso — для квантовой химии) производительность A64FX практически совпадала с сервером с Xeon, а на двух других приложениях была существенно выше.

В этой части обзора далее рассматриваются приложения и отвечающие им миниприложения, хотя производительность последних в сложных областях HPC может не давать хорошую оценку производительности всего приложения. Но для начала полезно проиллюстрировать общую ситуацию с доступными для A64FX приложениями на примере приложений вычислительной химии (см. ниже таблицу 7). Во-первых, ряд приложений, упоминаемых как доступные для работы на A64FX, просто портированы на AArch64, и не имеют поддержки SVE. А, например, разработчики квантовохимической программы ABINIT еще работали над ее тестированием на A64FX, и вероятно, что на рисунке 9 имеется в виду работающая на A64FX японская программа ABINIT-MP (см. также [13]), реализующая расчеты только для очень больших молекул специализированным приближенным методом FMO, к которой и относятся данные, представленные на этом рисунке.

Таблица 7. Программы вычислительной химии, обеспечивающие работу на A64FX

Программный комплекс, версия	Год выпуска с ARM	URL ¹	Тесты производительности
Квантовая химия			
Gaussian 16, Rev. C01	2021	https://gaussian.com/relnotes/	—
Gamess-US, R1	2021	https://www.msg.chem.iastate.edu/gamess/versions.html	—
ABINIT, 9 ²	2021		[38]
CASTEP, 18.1.0	2020		+
VASP ²	2019		—
NWChem, от 6.8 ²	2018	https://gitlab.com/arm-hpc/packages/-/wikis/packages/nwchem ⁴	—
CP2K, 8.2 ²	2021		
Quantum Espresso ⁵		https://github.com/fujitsu/oss-patches-for-a64fx	
SALMON, v.2.0.1	2020	http://salmon-tddft.jp/download/Manual_SALMON-v.2.0.1_simple_20210129.pdf	[38]
NTChem	2019	https://gitlab.com/arm-hpc/packages/-/wikis/packages/ntchem	+ ⁶
Молекулярная динамика			
GROMACS	2021	https://manual.gromacs.org/2021/release-notes/2021/major/highlights.html	+
LAMMPS		https://github.com/fujitsu/oss-patches-for-a64fx	+
NAMD ²	2019		—

¹Приведены только URL для версий, поддерживающих A64FX;²для AArch64;³в [38];⁴Для A64FX: https://static.linaro.org/.../TheFirstSVEEnabledArmProcessor_A64FXandBuildingupArmHPCEcosystem5.pdf⁵сделано Fujitsu;⁶NTChem-MINI, <https://github.com/fiber-miniapp/ntchem-mini>;

Из других данных о производительности, которые можно отнести к области квантовой химии, следует указать расчет TiN по CASTEP 18.1.0 (см. рисунок 10) [89, с. 18], где производительность на A64FX/2,2 ГГц сопоставлена с двухпроцессорными серверами: с Xeon 8260M/2,4 ГГц (всего — также 48 ядер) и с ThunderX2/2,2 ГГц (всего — 64 ядра). Данные на вертикальной оси отражают производительность, и видно,

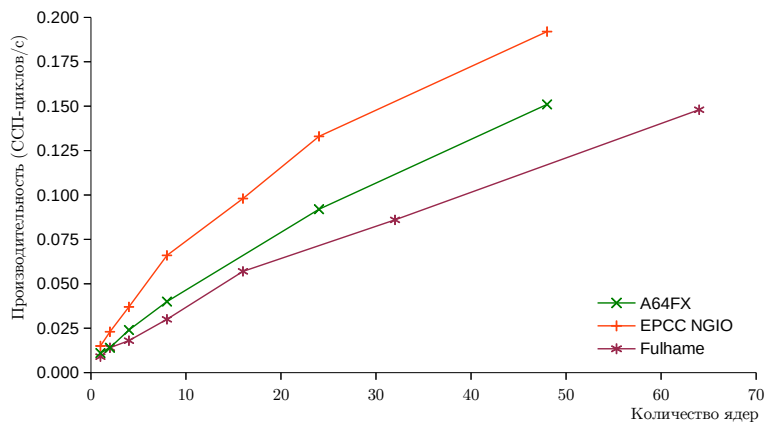


Рисунок 10. Сравнительная производительность A64FX в CASTEP

что A64FX явно опережает ThunderX2 и отстает от Cascade Lake. В таблице 8 приведена производительность этих двухпроцессорных серверов по отношению к A64FX. Но надо иметь в виду, что CASTEP

Таблица 8. Относительная производительность двухпроцессорных серверов с другими процессорами по отношению к серверу с A64FX

Сервер с:	minikab ¹	nekbone	CASTEP
A64FX	1,0	1,0	1,0
Xeon 8260M	~ 0,9	0,3	1,3
ThunderX2	~ 0,5	0,4	0,97

¹На одном процессорном ядре

ориентирован на исследования твердых тел и работу в базисе плоских

волн, где расчеты лимитируются БПФ, а в квантовой химии обычно работают в базисе гауссовских орбиталей.

Другим квантовохимическим приложением, работающим на A64FX, является реализующая метод TDDFT программа SALMON, которая, как и ABINIT выше, существенно опережает по производительности двухпроцессорный сервер с Xeon 8268 (имеющий также 48 ядер, 2,9 ГГц). Однако TDDFT в химии все-таки не является массово применяемым, как собственно DFT. А про проблемы с пропускной способностью памяти при работе с TDDFT говорилось еще до появления A64FX, в [99] — когда авторы работали на К-компьютере.

Более естественным для квантовых химиков тестировавшим-ся на A64FX приложением является возможно NTChem [50, 100]; тестировалось, точнее, мини приложение *NTChem-mini*^{URL}, которое является частью NTChem для расчетов методом RI-MP2 с учетом электронной корреляции, в котором применяется распараллеливание в MPI и XcalableMP, двигающееся ныне в PGAS-направлении [101]. Сопоставление производительности на NTChem-mini (на Fugaku) с К-компьютером в зависимости от числа узлов имеется в [101], при этом MPI давал практически ту же производительность, что и распараллеливание на XcalableMP. Полезно также отметить, что для расчетов энергии и ее градиента в MP2 на ARM-системах задолго до появления A64FX была найдена важной пропускная способность памяти [9].

Распараллеленное в OpenMP миниприложение для метода быстрых мультиполей (miniFMM) отнесено здесь к вычислительной химии, поскольку этот метод может применяться и для больших молекулярных систем (хотя из-за общей математики может работать и на уровне галактик). В [83] найдено, что узел на Fugaku A64FX/2,0 ГГц сильно опережал сервер с двумя 28-ядерными ThunderX2/2,0 ГГц, но заметно уступал по производительности серверу с двумя 12-ядерными Xeon Skylake/2,6 ГГц (вероятно, Xeon 6126) вплоть до 24 нитей, а на 48 нитей уже опережал из-за большего числа процессорных ядер.

К задачам вычислительной химии относится также приложение BUDE (Bristol University Docking Engine), также распараллеленное в OpenMP, где без применения квантовой механики рассчитывается энергия взаимодействия протеина с лигандом. Fugaku A64FX/2,0 ГГц в сравнении с указанными в предыдущем абзаце серверами по производительности выглядел аналогично miniFMM: сильно опережал ThunderX2, и до 24 нитей существенно уступал по производительности

серверу со Skylake, превосходя на 48 нитях [83]. В [102] рассмотрена производительность миниприложения на основе BUDE (miniBUDE).

Приложения молекулярной динамики относятся к вычислительно интенсивной группе НРС, и эффективно работают на GPU.

Подробный анализ расчетов молекулярной динамики по программе LAMMPS и способов возможного поднятия производительности на A64FX/2,2 ГГц, в том числе с использованием векторизации и разных вариантов распараллеливания, проведен в [103]. Для другого знаменитого приложения молекулярной динамики, GROMACS, ориентированного на биомолекулярные системы, использование C++ 17 отвергло возможность использования компиляторов Cray 10.0.1 и ARM 20.3, и применялся gcc 10.2.0. На A64FX/1,8 ГГц GROMACS (для системы из 87 тысяч атомов) достигло 20,2 нс/день, а двухпроцессорный вариант с Xeon 8160 (всего 48 ядер) 75,6 нс/день. Причины такого большого выигрыша в производительности у Skylake не выяснены; возможно, этому способствовало отсутствие векторизации элементарных функций в gcc [39].

В заключение части про приложения по вычислительной химии рассмотрим еще данные (таблицы 9 из [34]), которые немного иллюстрируют возможности использования разных режимов энергопотребления в A64FX, в том числе и для приложений в области биоинформатики. Здесь видно, что убыстренный режим по отношению

Таблица 9. Производительность (П) и мощность (М) относительно нормального режима (N) энергопотребления на приложениях вычислительной химии и биоинформатики

Приложение	B		N+E		B+E	
	П	М	П	М	П	М
GENESIS	1,09	1,20	1,0	0,8	1,09	0,96
Genomon	1,10	1,17				
NTChem	1,08	1,21	0,57	0,69	0,62	0,83
RSDFT	1,06	1,20	0,71	0,8	0,77	0,9

B — ускоренный режим; E — эко-режим;

GENESIS — приложение молекулярной динамики для биомолекулярных систем (<https://www.r-ccs.riken.jp/>);

Genomon — приложение биоинформатики (<https://genomon.hgc.jp/>);

RSDFT — квантовохимическое приложение.

к нормальному всегда давал ускорение производительности, однако для приложения GENESIS оптимально сочетание убыстренного и

эко-режимов, не дающее заметное уменьшение производительности. А для квантовохимических приложений включение эко-режима давало большое уменьшение производительности.

Среди приложений по вычислительной химии одна из основных частей — квантовая химия, где квантовая механика применяется к химическим объектам, но на A64FX проводились и чисто квантовомеханические расчеты, не имеющие отношения к квантовой химии.

В [64] с использованием BLIS-библиотеки проведены расчеты по усовершенствованной методике авторов — на A64FX, Xeon 8260 и EPYC 7702, но напрямую производительность между разными процессорами не сопоставлялась. В [104] в приложении DCA++ для расчетов квантовым методом Монте-Карло с распараллеливанием на OpenMP проведено сопоставление A64FX и ThunderX2 с применением ARMclang 20.3 и ArmPL 20.3. Здесь использовались и SIMD-директивы OpenMP, но с точки зрения сравнительной производительности показано только очевидное опережение A64FX (в разы) относительно ThunderX2.

Приведенные данные свидетельствуют о том, что современные процессоры x86-64 (данных сравнения производительности A64FX с более новыми Xeon Ice Lake нет) часто опережают A64FX в задачах вычислительной химии. Традиционные оценки, что для квантовохимических расчетов нужно в типичном случае 2 ГБ памяти на процессорное ядро, для многоядерного A64FX неприемлемы, и требуют дополнительного анализа. По другой грубой оценке, хорошего для типового квантовохимического расчета отношения 1 Гбайт/с в памяти на 1 GFLOPS, процессор A64FX выглядит привлекательнее других. Но смотреть пользователь все равно будет на работоспособность и производительность приложения.

Программные средства BWA-MEM2 для решения задач геномики, отнесенные к HPC, были портированы на A64FX в [41]; ; однако достигнутая производительность A64FX на Fugaku оказалась ниже, чем на Xeon 8160. Хотя пропускная способность в A64FX гораздо выше, здесь могли сказаться большие задержки работы с памятью в A64FX [41].

Как и в квантовой химии, квантовая механика используется и в КХД, производительность приложений которой активно изучалась на A64FX с расчетами на FX700 и FX1000. В использующейся решеточной модели КХД важным является умножение матрицы на вектор [43, 87, 105]. Производительность SpMV на A64FX была рассмотрена выше, здесь речь пойдет о производительности в решеточной КХД. Здесь важно отметить, что использование A64FX для

расчетов КХД предполагалось весьма перспективным, и был создан специальный проект QPACE 4 создания суперЭВМ для параллельных расчетов КХД с базированием на A64FX, в котором участвовала и Cray. Эта система, использующая FX700, где 64 узла соединены через Infiniband EDR, начала работу в 2020 году [54].

В [43] в рамках приложения GRID (C++, с распараллеливанием OpenMP+MPI) для решеточной КХД была исследована производительность ядра Domain Wall (DW). С использованием чередующейся формы хранения данных RIRI (R-действительная, I-мнимая часть) производительность A64FX (из Fugaku) составила около 440 GFLOPS. Но более эффективной была найдена форма RRII; здесь было достигнуто 182 GFLOPS на CMG (на 12 ядрах), и 712 GFLOPS на всем A64FX, на 12% больше, чем с RIRI.

В [43] исследовано также влияние выбора различных вариантов управления потребляемой мощностью A64FX для RIRI и RRII. Энергопотребление с RIRI было выше, чем с RRII. В отличие от RIRI, при работе с RRII можно было уменьшить энергопотребление за счет уменьшения тактовой частоты (с 2,2 до 2,0 ГГц) и включения эконо-режима, при этом производительность существенно не снижалась. В [87] для аналогичных DW-расчетов было найдено возможным понижение энергопотребления на таком A64FX на 18%.

Данные достигаемой производительности A64FX для различных размеров решетки (по одной из координат) по схеме RRII были сравнены с производительностью на V100 и двухпроцессорном сервере с Xeon 9242 (всего 96 процессорных ядер), но в них использовалась только схема RIRI. С RRII A64FX практически всегда был самым быстрым; с RIRI в зависимости от размера решетки иногда был быстрее A64FX, иногда — V100, а Xeon Cascade Lake им существенно уступал. Но для достижения высокой производительности для задач КХД на A64FX в [43] понадобилась модернизация кода.

В [43, 87] отмечено уменьшение достигаемой производительности решеточной КХД на A64FX из-за больших задержек команд и недостаточности ресурсов для OoO, в том числе небольшого размера буфера ROB.

Эти данные по производительности на Fugaku с A64FX/2,2 ГГц были сопоставлены с производительностью на суперЭВМ QPACE 4 с A64FX/1,8 ГГц [53, 54]. На QPACE 4 были проведены расчеты с использованием SP и DP, но приводимые данные о производительности относятся к DP. Со схемой RIRI на ядре Wilson Dslash (WD) было получено 376 GFLOPS на узел, и исследована зависимость производительности с SP и DP в зависимости от числа узлов (до 32).

В ядре DW (Domain Wall) было получено 475 GFLOPS на узел, и исследована такая же зависимость от числа узлов, как в WD. Схема RRII также давала здесь более высокую производительность, чем RIRI; на ядре DW была получена на порядка 20% более высокая производительность, чем с RIRI (исследованы разные варианты решеток).

Хотя тактовая частота A64FX на QPACE 4 понижена в 1,22 раза относительно Fugaku, полученная производительность понизилась до порядка 25%. Однако на QPACE 4 не были доступны некоторые возможности A64FX, которые применялись на Fugaku, в том числе секторный кэш или эко-режимы [54]. В [53, 54] отмечено, что оптимизация кодов этих ядер требовала ручной работы, а в [54] — отсутствие за прошедший год прогресса в работе компиляторов с SVE.

Для решеточной КХД GRID — не единственное приложение для A64FX; в [75] указаны еще Bridge++, BQCD и LDDHMC. Кроме того, имеется специальная разрабатываемая Fujitsu библиотека QWS [106] для задач КХД. Методы оптимизации, используемые в QWS, являются общими, за исключением использования ACLE и средств работы с TofuD [76].

С использованием QWS и BiCGstab (стабилизированный метод бисопряженных градиентов) с распараллеливанием OpenMP+MPI на 147456 узлах Fugaku в убыстренном режиме (с частотой 2,2 ГГц) без включения эко-режима была получена указанная выше в разделе в разделе 3.1 SP-производительность 102 PFLOPS и энергоэффективность 5 GFLOPS/Вт [75]. Детальнее информация об этом расчете и данные о производительности, начиная от двух задействованных CMG, приводится в [76].

В [105] для решеточной КХД производительность их объектно-ориентированного кода, и при использовании BiCGstab со смешанной точностью была сравнена в том числе на A64FX и V100; при этом их код давал лучше производительность и ее масштабирование. Здесь была также отмечена необходимость адаптации кода для конкретной используемой архитектуры с целью достижения максимальной производительности.

Следующая группа приложений, производительность которых была получена на A64FX, относится к релятивистской механике и астрофизике. В [107] производительность кластера с узлами из HPE CS500 исследована на приложении релятивистской магнитогидродинамики — ВНАС 3 (распараллеливание MPI+OpenMP). Однако достигнутая производительность оказалась примерно в 3 раза меньше,

чем на двухпроцессорном сервере с Xeon 8174 (48 ядер на сервер). В [39] производительность на A64FX в Ookami изучалась в приложении FLASH, ориентированном на использование в разных областях физики, в том числе астрофизики, однако проблемы с компиляторами, в том числе в обеспечении генерации кодов с поддержкой SVE, не дали получить интересные результаты.

В [108] производительность A64FX (на Apollo 80, в Ookami) изучена в рамках приложения VPIC 2.0, использующего средства Kokkos и ориентированного на решение задач релятивистской физики плазмы (они также актуальны для астрофизики). На рисунке 11, основанном на [108, рис. 10] производительность A64FX/1,8 ГГц сопоставлена с достигаемой в EPYC (Zen 2) и Xeon Cascade Lake. Процессоры

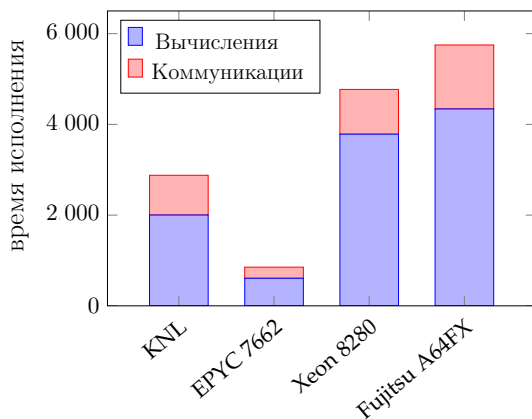


Рисунок 11. Сопоставление производительности на VPIC в кластерах: вычисления и коммуникации

x86-64 использовались в кластерах с двухпроцессорными узлами, соединенными Infiniband HDR. Процессорное время на восьми EPYC 7742 было раз в 6 меньше, чем у восьми A64FX, и время коммуникаций на EPYC-системе также было в разы меньше. 8 процессоров Xeon 8280 также существенно превосходили восемь A64FX по этим параметрам. Из-за неэффективной реализации векторизации в Kokkos только EPYC давал сопоставимую производительность с GPU V100 и A100 [108].

Большое количество приложений, для которых стали доступными данные об их производительности на A64FX, не используют ни квантовую, ни релятивистскую физику. Они охватывают разные области, например, CFD и прогноз погоды.

Так, в [39, 52, 58] на A64FX/1,8 ГГц получены данные о производительности на известном приложении предсказания погоды SWIM, распараллеленном в OpenMP. Исследование производительности на A64FX (на Fugaku) знаменитого приложения Nek5000 для турбулентных атмосферных явлений проведено в [56]. Исследована производительность Nek5000 в большом диапазоне числа узлов Fugaku. В рамках теста BP3 из CEED, использующего аналогичную математику, производительность A64FX сопоставлена с V100 на Summit и Xeon Gold 6130/2,1 ГГц (16 ядер). У A64FX и V100 производительность иногда была близка, а иногда V100 опережал A64FX в 2-3 раза, а Skylake всегда отставал. В [89] на миниприложении Nekbone для Nek5000 для A64FX/2,2 ГГц была получена производительность 176 GFLOPS, больше, чем на двухпроцессорных серверах: с Xeon 8260M (также всего 48 ядер/2,4 ГГц) — 127 GFLOPS, а с ThunderX2 (всего 64 ядра/2,2 ГГц) — 122 GFLOPS.

Производительность таких же вычислительных систем была сопоставлена на другом CFD-приложении, COSA, в [89], но здесь эти вычислительные системы выступали в качестве узлов кластеров с межсоединениями TofuD — для A64FX, Infiniband EDR — для ThunderX2, OmniPath — для Xeon 8260M. На рисунке 12 согласно [89, с. 15] приведены графики зависимости времени расчета от числа используемых узлов, система с ThunderX2 отмечена как Fulhame, а с Xeon — как NGIO. A64FX здесь требует меньше времени для расчета

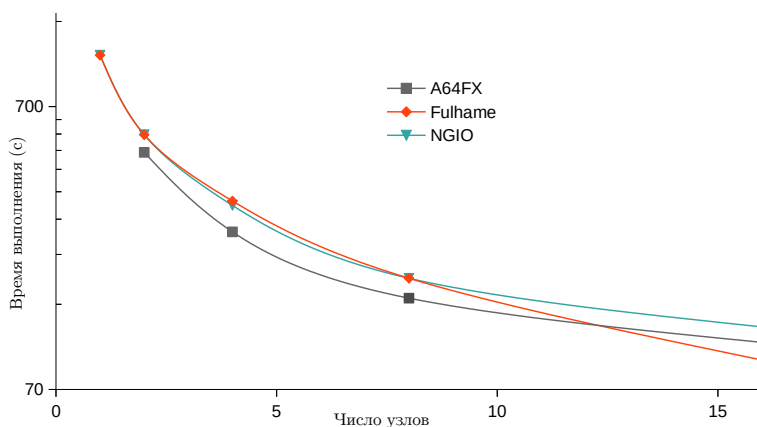


Рисунок 12. Производительность в COSA

при небольшом числе узлов. Но в общем производительности этих вычислительных систем при работе с COSA достаточно близки. Другое

CFD-приложение, ShallowWaters.jl (для циркуляции жидкости), созданное в [109] на A64FX для работы с половинной точностью, ускорило расчет в 3,6 раза по сравнению с DP, что может стать актуальным для моделирования погоды и климата в будущем. Для CFD на A64FX было получено много данных о производительности. Так, в [83] полученные данные по производительности A64FX/2,0 ГГц на приложениях LULESH и CloverLeaf (распараллеленных в MPI+OpenMP) сопоставлены с производительностью на двухпроцессорных серверах: с Xeon Skylake с 24 ядрами (суммарно) с частотой 2,6 ГГц (вероятно, Xeon 6126) и с 28-ядерными ThunderX2/2,0 ГГц. В LULESH A64FX уступал по производительности и ThunderX2, и Skylake — в том числе потому, что в этом приложении важна задержка памяти; аналогичные данные имеются в [38]. По производительности в прокси-приложении LULESH A64FX/1,8 ГГц существенно уступал двухпроцессорному серверу с Xeon 6130 (32 ядра на сервер) [59].

В CloverLeaf при разном числе OpenMP-нитей и процессов MPI процессор A64FX всегда опережал ThunderX2, и обычно опережал и Skylake [83]. По данным [38], сопоставимость по производительности A64FX с CloverLeaf достигалась только при работе на двух двухпроцессорных серверах со Skylake. Такое преимущество A64FX — следствие того, что производительность мини-приложения CloverLeaf лимитируется пропускной способностью памяти [34].

На A64FX/2,2 ГГц изучена производительность еще одного приложения, относимого к CFD — OPenSBLI. Его оригинальность для НРС связана с использованием исходного текста на Python, который для решения конкретных дифференциальных уравнений, записанных в нотации Эйнштейна, генерирует код на C, в котором будет применяться распараллеливание MPI+OpenMP [89]. Выше в таблице 6 приведены данные о получаемой производительности в кластерах, содержащих до 8 узлов: с A64FX (с соединенными TofuD узлами) и кластеров из двухпроцессорных серверов: с Xeon 8260M (всего 48 ядер/2,4 ГГц) — с межсоединением OmniPath, и с ThunderX2 (всего 64 ядра/2,2 ГГц) — с межсоединением Infiniband EDR. Здесь при любом числе узлов кластер с A64FX в разы отстает по производительности от альтернативных кластеров с таким же числом узлов; предварительный анализ причин этого дал предположение о возможной модификации кода OPenSBLI с целью повышения производительности на A64FX [89].

На базе кода FLAG, используемого для задач радиационной CFD, создано мини-приложение PENNANT, решающее задачи физики сеток [110]. В [39, 52, 58] исследована производительность в PENNANT

на A64FX/1,8 ГГц при OpenMP-распараллеливании (с числом ядер до 48).

В ряде работ [39, 52, 58, 60] имеются данные сравнения производительности A64FX с другими процессорами и с GPU в мини-приложении *minimod* для задач сейсмического моделирования [60]. В [39, 52, 58] приводятся данные о достигаемом на A64FX ускорении при распараллеливании на OpenMP в зависимости от числа используемых процессорных ядер. В [60] использовано MPI-распараллеливание, и проанализирована величина достигаемого ускорения в зависимости от числа ядер (до 48) на A64FX и EYUC 7702 (Roma, 64 ядра); на A64FX ускорение с ростом числа ядер становилось гораздо больше, чем на Roma. А что касается производительности *minimod*, то на A64FX расчеты выполнялись в разы быстрее, чем на Roma, а сервер на AMD иногда существенно опережал, а иногда отставал от двухпроцессорного сервера с Xeon 5112 (всего 24 ядра). A64FX всегда опережал x86-64, но отставал по производительности от V100 [60].

В [38, 71] представлены данные о производительности на A64FX из Fugaku распараллеленного на MPI+OpenMP мини-приложения Tealeaf, которое решает линейное уравнение теплопроводности. Полученная при этом на A64FX производительность (исследована также зависимость от числа OpenMP-нитей и MPI-ранга) была больше, чем у двухпроцессорного сервера с Xeon Skylake (всего 24 ядра/2,6 ГГц) и у двухпроцессорного сервера с ThunderX2 (всего 56 ядер/2,0 ГГц). При этом полученная производительность у A64FX была в два раза больше, чем у двух двухпроцессорных серверов со Skylake. Но в этом мини-приложении производительность лимитируется пропускной способностью памяти [38].

Только что рассмотренные приложения/мини-приложения с данными по производительности на A64FX отличались тем, что в них не применяется ни квантовая, ни релятивистская механика. Ниже приводятся данные о других изученных на A64FX и относительно реже используемых приложениях. Например, в [84] на A64FX использовались задачи адаптивной оптики, которые должны решаться в реальном времени. Производительность A64FX/2,2 ГГц была сопоставлена в том числе с двухпроцессорными серверами с Xeon 6248 (всего 40 ядер/2,5 ГГц), с EYUC 7702 (всего 128 ядер/2,2 ГГц) и с GPU A100, V100 и AMD MI100. Время расчета на A64FX было сопоставимо с потребовавшимся на этих GPU (и на NEC SX-Aurora TSUBASA), и обычно существенно меньше, чем на серверах с x86-64. Время расчета здесь сильно зависит от эффективности реализации умножения матрицы на вектор.

В [89] на кластерной системе с A64FX/2,2 ГГц в узлах, соединенных TofuD, изучена производительность миниприложения minikab (Mini Krylov ASiMoV Benchmark), использующего метод сопряженных градиентов и распараллеленного в MPI+OpenMP, в сравнении с кластером из двухпроцессорных серверов с 32-ядерными процессорами ThunderX2/2,2 ГГц (с Infiniband EDR). Использовалось до 6 узлов с ThunderX2 и до 8 узлов — с A64FX (до 384 ядер в каждом кластере). При равном числе процессорных ядер кластер с A64FX существенно превосходил по производительности кластер с ThunderX2. Это соответствует также представленным в [89] сравнительным оценкам производительности minikab на одном процессорном ядре: в A64FX — в два раза быстрее, чем в ThunderX2 и на 7% быстрее Xeon 8260M.

В связи с широким распространением в последнее время задач ИИ, которые могут давать потенциально более широкий рынок, чем традиционные HPC, производительность A64FX изучалась и на таких задачах. Для ИИ чаще актуальны расчеты с половинной точностью, особенно с bfloat16 — с теми же 8 битами для показателя степени, как в SP — что нехарактерно для традиционных HPC с DP; поэтому данные о такой производительности рассматриваются совсем коротко и в конце данного раздела 3.3 обзора.

Данные о производительности Fugaku в известном в мире суперЭВМ тесте HPL-AI приведены выше в разделе в разделе 3.1. А вообще ИИ применялся и в задаче эмуляции самого A64FX — проверялись полученные задержки выполнения команд [44].

В [111] получены данные о производительности на Fugaku для остаточной нейронной сети ResNet-50 для классификации изображений, а в использующем глубокое обучение приложении Chainer (на Python — с использованием NumPy) получено линейное масштабирование производительности (числа обработанных изображений в секунду) при увеличении числа узлов Fugaku до 8. ResNet-50 вообще теперь применяется, например, и для задач обнаружения COVID-19 на рентгеновских снимках [112]. В [113] производительность A64FX/2,2 ГГц на ResNet-50 сопоставлена с производительностью двухпроцессорного сервера с Xeon 8268 (всего 48 ядер/2,9 ГГц). По производительности A64FX уступал лишь немного, но энергоэффективность в A64FX была в 2,8 раз выше, чем у Xeon.

На основании данных о производительности, приведенных выше в разделах 2 и 3 ясно, что процессор Fujitsu практически всегда существенно выигрывал в производительности у ThunderX2, что было вполне ожидаемым. Все данные о производительности A64FX в тестах и приложениях говорят о том, что A64FX получает преимущество

над серверными процессорами x86-64 как правило тогда, когда для достижения высокой производительности необходимо иметь высокую пропускную способность памяти. А в большей части приложений и тестов более высокую производительность получают современные x86-64 процессоры Intel и AMD. GPU в естественных для них областях применения имеют обычно производительность сильно больше, чем A64FX. Кроме того, для достижения высокой производительности A64FX нередко требуется модифицировать исходные коды программ.

Заключение









Данные о производительности и энергоэффективности A64FX свидетельствуют, что использование A64FX в суперЭВМ пока будет скорее расширяться. На основании данных об используемых в суперЭВМ ARM-процессорах ThunderX2 и A64FX можно ожидать, что гегемония x86-64 процессоров в HPC будет явно уменьшаться — об этом уже фактически говорилось, например, в документах европейской HPC-инфраструктуры PRACE. Однако данные микроархитектуры показывают такие недостатки A64FX по сравнению с современными серверными процессорами x86-64, как увеличенные задержки в ряде инструкций и при работе с памятью, и недостаточность OoO-ресурсов, с которой можно отчасти справляться. Кроме того, средства разработки программ для A64FX пока нуждаются в усовершенствовании. Простая трансляция исходного текста компиляторами для A64FX без переработки исходного текста в целях оптимизации может давать не ту производительность, что ожидалась. Преимущества в производительности на A64FX получаются, когда выигрыш обусловлен кардинально более высокой пропускной способностью HBM2.


















На тестах и приложениях A64FX, пожалуй, чаще проигрывает в производительности процессорам Xeon начиная со Skylake (а с Ice Lake, где должна проявляться наибольшая производительность, сопоставлений вообще нет) и EPYC Roma/Milan. Представляется, что в анализ производительности надо бы явно включать и данные о стоимости и реально потребляемой электроэнергии, то есть также стоимостного показателя. И сейчас нельзя определить, какой из AArch64-процессоров будет в ближайшее время ведущим в процессе оттеснения x86-64 из абсолютного лидерства в HPC: Fujitsu A64FX, SiPearl Rhea, Nvidia Grace или какой другой (например, Huawei Kunpeng 930 — если скоро начнется его производство).


Имеются публикации о проведенных в частности в RIKEN исследований о возможных путях развития A64FX, для чего использовалась

эмуляция процессора (Gem5 с добавленной SVE-эмуляцией). В [37] для 1024-битного SVE показана возможность ускорения производительности при снижении энергопотребления. А в [5] исследовались возможность простого увеличения числа ядер и увеличение длины SVE-векторов до 2048 бит при использовании технологий 5 и 3 нм. Можно предположить, что это будет возможным главным направлением работы Fujitsu в будущем.

Список литературы

- [1] A. Tekin, Tuncer Durak A., Piechurski C., Kaliszan D., Aylin Sungur F., Robertsén F., Gschwandtnr P. *State-of-the-art and trends for computing and interconnect network solutions for HPC and AI*, Technical report: PRACE.– 2021.– 38 pp.  [↑](#)_{64, 72, 73}
- [2] Liabotis I. *EINFRA-4-2014: Pan-European high performance computing infrastructure and services*, PRACE-4IP-EINFRA-653838: PRACE.– 2017.– 71 pp.  [↑](#)_{64, 66}
- [3] Edwards C. *Moore's Law: what comes next?* // Communications of the ACM.– 2021.– Vol. **64**.– No. 2.– pp. 12–14.  [↑](#)₆₄
- [4] Domke J., Vatai E., Drozd A., Chen P., Oyama Y., Zhang L., Salaria S., Mukunoki D., Podobas A., Wahib M., Matsuoka S. *Matrix engines for high performance computing: A paragon of performance or grasping at straws?*, 2021 IEEE International Parallel and Distributed Processing (IPDPS) (17–21 May 2021, Portland, OR, USA).– 2021.– pp. 1056–1065.  [↑](#)_{64, 73, 95}
- [5] Arima E., Kodama Y., Odajima T., Tsuji M., Sato M. *Power/Performance /Area Evaluations for Next-Generation HPC Processors using the A64FX Chip*, 2021 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS) (14–16 April 2021, Tokyo, Japan).– 2021.– pp. 1–6.  [↑](#)_{64, 95, 98, 120}
- [6] M. S. Gordon, Barca G., S. S. Leang, Poole D., A. P. Rendell, J. L. Galvez Vallejo, Westheimer B. *Novel computer architectures and quantum chemistry* // The Journal of Physical Chemistry A.– 2020.– Vol. **124**.– No. 23.– pp. 4557–4582.  [↑](#)_{64, 71}
- [7] Calore E., Gabbana A., S. F. Schifano, Tripiccone R. *ThunderX2 performance and energy-efficiency for HPC workloads* // Computation.– 2020.– Vol. **8**.– No. 1.– pp. 20.  [↑](#)_{64, 65, 85}
- [8] Tiwari A., Keipert K., Jundt A., Peraza J., S. S. Leang, Laurenzano M., M. S. Gordon, Carrington L. *Performance and energy efficiency analysis of 64-bit ARM using GAMESS* // *Proceedings of the 2nd International Workshop on Hardware-Software Co-Design for High Performance Computing, Co-HPC '15* (15 November 2015, Austin, Texas, USA), New York:ACM.– 2015.– ISBN 978-1-4503-3992-6.– 10 pp.  [↑](#)₆₄
- [9] Keipert K., Mitra G., Sunriyal V., S. S. Leang, Sosonkina M., A. P. Rendell, M. S. Gordon *Energy-efficient computational chemistry: Comparison of x86*














- and ARM systems* // Journal of Chemical Theory and Computation.– 2015.– Vol. **11**.– No. 11.– pp. 5055–5061.  ^{↑64, 109}
- [10] Saastad O. W., Kapanova K., Markov S., Morales C., Shamakina A., Johnson N., Krishnasamy E., Varrette S. *Best Practice Guide Modern Processors*: PRACE.– 2020.– 109 pp.  ^{↑64, 66, 69, 71, 75, 76}
- [11] Антонов А. С., Афанасьев И. В., Воеводин Вл. В. *Высокопроизводительные вычислительные платформы: текущий статус и тенденции развития* // Вычислительные методы и программирование.– 2021.– Т. **22**.– № 2.– с. 135–177.  ^{↑64, 66}
- [12] Xia J., Cheng C., Zhou X., Hu Y., Chun P. *Kunpeng 920: The first 7nm chiplet-based 64-Core ARM SoC for cloud services* // IEEE Micro.– 2021.– Vol. **41**.– No. 5.– pp. 67–75.  ^{↑64, 66, 68}
- [13] Dongarra J. *Report on the Fujitsu Fugaku system*, Tech Report No ICL-UT-20-06: University of Tennessee, Innovative Computing Laboratory.– 2020.– 18 pp.  ^{↑64, 74, 91, 94, 97, 104, 106}
- [14] Zhang W., Jiang Z., Chen Z., Xiao N., Ou Y. *NUMA-Aware DGEMM based on 64-bit ARMv8 multicore processors architecture* // Electronics.– 2021.– Vol. **10**.– No. 16.– pp. 1984.  ^{↑64, 68, 71}
- [15] Фролов В., Галактионов В., Санжаров В. *RISC-V: стандарт, изменивший мир микропроцессоров* // Открытые системы. СУБД.– 2020.– № 2.– с. 30–34.  ^{↑64}
- [16] Кузьминский М. *Power10: возрождение RISC* // Открытые системы. СУБД.– 2021.– № 3.– с. 10–12.   ^{↑65, 78}
- [17] Jiang L., Yang C., W. Ma *Enabling highly efficient batched matrix multiplications on SW26010 many-core processor* // ACM Transactions on Architecture and Code Optimization.– 2020.– Vol. **17**.– No. 1.– pp. 1–23, 2.  ^{↑65}
- [18] Кузьминский М. *Китайский процессорно-суперкомпьютерный путь* // Открытые системы. СУБД.– 2017.– № 1.– с. 8–11.   ^{↑65}
- [19] Кузьминский М. *ARM для HPC: время пришло?* // Открытые системы. СУБД.– 2020.– № 2.– с. 12–15.   ^{↑65}
- [20] Ouro P., Lopez-Novoa U., Guest M. F. *On the performance of highly-scalable Computational Fluid Dynamics code on AMD, ARM and Intel processor-based HPC systems* // Computer Physics Communications.– 2021.– Vol. **269**, 108105.  ^{↑65, 69, 74}
- [21] McIntosh-Smith S., Price J., Deakin T., Poenaru A. *A performance analysis of the first generation of HPC-optimized Arm processors* // Concurrency and Computation: Practice and Experience.– 2019.– Vol. **31**.– No. 16, e5110.  ^{↑65}
- [22] Stephens N., Biles S., Boettcher M., Eapen J., Eyole M., Gabrielli G., Horsnell M., Magklis G., Martinez A., Premillieu N., Reid A., Rico A., Walker P. *The ARM scalable vector extension* // IEEE Micro.– 2017.– Vol. **37**.– No. 2.– pp. 26–39.  ^{↑66}












- [23] Soria-Pardos V., Armejach A., Suárez D., Moretó M. *On the use of many-core Marvell ThunderX2 processor for HPC workloads* // The Journal of Supercomputing.– 2021.– Vol. **77**.– No. 4.– pp. 3315–3338.  [↑66](#)
- [24] Sugumar R. *ThunderX3 next-generation arm-based server*, 2020 IEEE Hot Chips 32 Symposium (HCS) (16–18 Aug., 2020, Palo Alto, CA, USA).– 2020.– pp. 1–19.  [↑66](#)
- [25] Sugumar R., Shah M., Ramirez R. *Marvell ThunderX3: next-generation arm-based server processor* // IEEE Micro.– 2021.– Vol. **41**.– No. 2.– pp. 15–21.  [↑66](#)
- [26] Gao W., Fang J., Huang C., Xu C., Wang Z. *Optimizing barrier synchronization on ARMv8 many-core architectures*, 2021 IEEE International Conference on Cluster Computing (CLUSTER) (7–10 Sept. 2021, Portland, OR, USA).– pp. 542–552.  [↑68](#)
- [27] All SPEC CPU2017 results published by SPEC.  [↑68](#)
- [28] McCalpin J. D. *HPL and DGEMM performance variability on the Xeon Platinum 8160 processor*, SC18: International Conference for High Performance Computing, Networking, Storage and Analysis (11–16 Nov. 2018, Dallas, TX, USA).– pp. 225–237.  [↑71](#)
- [29] Кружилов И. С., Кузьминский М. Б., Чернецов А. М., Шамаева О. Ю. *Базовые библиотеки линейной алгебры для высокопроизводительных расчетов* // Вестник МЭИ.– 2018.– № 6.– с. 87–95.   [↑71](#)
- [30] Álvarez-Farré X., Gorobets A., Trias F., Oliva A. *NUMA-aware strategies for the heterogeneous execution of SPMV on modern supercomputers*, 14th WCCM-ECCOMAS Congress 2020 (19–24 July 2020, Paris, France).– 10 pp.  [↑71](#)
- [31] Mahmoud M., Hoffmann M., Reza H. *Developing a new storage format and a warp-based SpMV kernel for configuration interaction sparse matrices on the GPU* // Computation.– 2018.– Vol. **6**.– No. 3.– pp. 45.  [↑71](#)
- [32] Zhang Y., Yang W., Li K., Tang D., Li K. *Performance analysis and optimization for SpMV based on aligned storage formats on an ARM processor* // Journal of Parallel and Distributed Computing.– 2021.– Vol. **158**.– pp. 126–137.  [↑71](#)
- [33] Afanasyev I., Lichmanov D. *Evaluating the performance of Kunpeng 920 processors on modern HPC applications* // PaCT 2021: Parallel Computing Technologies, International Conference on Parallel Computing Technologies, Lecture Notes in Computer Science.– vol. **12942**, Cham:Springer.– 2021.– ISBN 978-3-030-86359-3.– pp. 301–321.  [↑72](#)
- [34] Sato M., Ishikawa Y., Tomita H., Kodama Y., Odajima T., Tsuji M., Yashiro H., Aoki M., Shida N., Miyoshi I., Hirai K., Furuya A., Asato A., Morita K., Shimizu T. *Co-Design for A64FX manycore processor and “Fugaku”* // SC '20: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (9–19 November, 2020, Atlanta, Georgia, USA).– 2020.– ISBN 978-1-7281-9998-6.– pp. 1–15.  [↑74, 75, 76, 77, 78, 95, 97, 102, 106, 110, 116](#)

- [35] Okazaki R., Tabata T., Sakashita S., Kitamura K., Takagi N., Sakata H., Ishibashi T., Nakamura T., Ajima Y. // *Fujitsu Technical Review* 2020-03.– 2020.– 9 pp. [URL](#) ↑74, 75, 76, 77, 78
- [36] Perks O. *AEM and SVE*, International Workshop on HighPerformance Computing and Programming on Quantum Chemistry and Physics 2020 (HPCPQCP2020).– 2020.– 68 pp. [URL](#) ↑76
- [37] Odajima T., Kodama Y., Sato M. *Performance and power consumption analysis of ARM scalable vector extension* // *The Journal of Supercomputing*.– 2021.– Vol. **77**.– No. 6.– pp. 5757–5778. [doi](#) ↑76, 120
- [38] Sato M., Odajima T., Kodama Y. *Performance evaluation of the supercomputer “Fugaku” and A64FX manycore processor*, ScalA workshop 2020 (12th Nov, 2020).– 27 pp. [URL](#) ↑76, 93, 95, 105, 106, 107, 116, 117
- [39] Burford A., Calder A. C., Carlson D., Chapman B., CoŞkun F., Curtis T., Feldman C., Harrison R. J., Kang Y., Michalow-Icz B., Raut E., Siegmann E., Wood D. G., Deleon R. L., Jones M., Simakov N. A., White J. P., Oryspayev D. *Ookami: deployment and initial experiences*.– 2021. [arXiv](#) [2106.08987](#) ↑78, 80, 81, 84, 85, 88, 89, 90, 96, 97, 98, 102, 110, 114, 115, 116, 117
- [40] Huang et al H. *Shuhai: a tool for benchmarking HighBandwidth memory on FPGAs* // *IEEE Transactions on Computers*.– Vol. **71**.– No. 5.– pp. 1133–1144.– 12 pp. [doi](#) [URL](#) ↑78
- [41] Langerita Benitez R. *Evaluation of genome alignment workflows on HPC processors*, Master thesis: Universitat Politècnica de Catalunya.– 2021.– 69 pp. [URL](#) ↑78, 87, 98, 111
- [42] Alappat C., Laukemann J., Gruber T., Hager G., Wellein G., Meyer N., Wettig T. *Performance modeling of streaming kernels and sparse matrix-vector multiplication on A64FX*, 2020 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS) (12 Nov. 2020, Atlanta, GA, USA).– pp. 1–7. [doi](#) ↑78
- [43] Alappat C., Meyer N., Laukemann J., Gruber T., Hager G., Wellein G., Wettig T. *ECM modeling and performance tuning of SpMV and Lattice QCD on A64FX*.– 2021. [arXiv](#) [2103.03013](#) ↑82, 87, 88, 97, 98, 100, 101, 111, 112
- [44] Li L., Pandey S., Flynn T., Liu H., Wheeler N., Hoisie A. *SimNet: computer architecture simulation using machine learning*.– 2021. [arXiv](#) [2105.05821](#) ↑118
- [45] Schreier J. *Optimization of small matrix multiplication kernels on Arm*, Bachelor’s Thesis in Informatics: Technische Universität München.– 2021.– 47 pp. [URL](#) ↑78, 88, 96
- [46] Koo D., Lee J., Liu J., Byun E.-K., Kwak J.-H., Lockwood G. K., Hwang S., Antypas K., Wu K., Eom H. *An empirical study of I/O separation for burst buffers in HPC systems* // *Journal of Parallel and Distributed Computing*.– 2021.– Vol. **148**.– pp. 96–108. [doi](#) ↑78
- [47] Anzt H., Tsai Y. M., Abdelfattah A., Cojean T., Dongarra J. *Evaluating the performance of NVIDIA’s A100 Ampere GPU for sparse and batched computations*, 2020 IEEE/ACM Performance Modeling, Benchmarking and

- Simulation of High Performance Computer Systems (PMBS) (12 Nov. 2020, Atlanta, GA, USA).— pp. 26–38. [doi](#) ↑79, 95, 97, 100
- [48] Zhang L., Okamoto T., Ishii S., Hirai K., Sumimoto S., Gerofi B., Takagi M., Ishikawa Y. *OS enhancement in supercomputer Fugaku*, Fujitsu Technical Review.— 2020.— 7 pp. [URL](#) ↑80
- [49] Domke J., Matsumura K., Wahib M., Zhang H., Yashima K., Tsuchikawa T., Tsuji Y., Podobas A., Matsuoka S., *OS double-precision FPUs in High-Performance Computing: an embarrassment of riches?* 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS) (20–24 May 2019, Rio de Janeiro, Brazil).— pp. 78–88. [doi](#) ↑80
- [50] Domke J. *A64FX—Your Compiler You Must Decide!*.— 2021. [arXiv](#) [2107.07157](#) ↑81, 82, 83, 84, 89, 103, 109
- [51] Ishikawa K. I., Kanamori I., Matsufuru H., Miyoshi I., Mukai Y., Nakamura Y., Nitadori K., Tsuji M. *102 PFLOPS lattice QCD quark solver on Fugaku*.— 2021. [arXiv](#) [2109.10687](#) ↑81
- [52] Michalowicz B., Raut E., Kang Y., Curtis T., Chapman B., Oryspayev D. *Comparing OpenMP implementations with applications across A64FX platforms*.— 2021. [arXiv](#) [2107.10346](#) ↑81, 84, 85, 89, 115, 116, 117
- [53] Meyer N., Georg P., Solbrig S., Wettig T. *Grid on QPACE 4*, The 38th International Symposium on Lattice Field Theory (26–30 Jul., 2021).— 1 pp. [URL](#) ↑81, 83, 84, 96, 98, 112, 113
- [54] Meyer N. *Grid Lattice QCD framework on A64FX*, R-CCS seminar (online) (June 30, 2021, University of Regensburg, Regensburg, Germany).— 2021.— 29 pp. [URL](#) ↑81, 83, 84, 98, 112, 113
- [55] Arm Ltd *SVE compilers and libraries*, Arm SVE Hackathon on Ookami, February 2021.— 2019.— 41 pp. [URL](#) ↑81, 82
- [56] T. Kolev, Fischer P., Austin A. P., Barker A. T., Beams N., Brown J., Camier J.-S., Chalmers N., Dobrev V., Dudouit Y., Ghaffari L., Kerkemeier S., Lan Y.-H., Merzari E., Min M., Pazner W., Rathnayake T., Shephard M. S., Siboni M. H., Smith C. W., Thompson J. L., Tomov S., Warburton T. *High-order algorithmic developments and optimizations for large-scale GPU-accelerated simulations*, ECP Milestone Report WBS 2.2.6.06, Milestone CEED-MS36: US Department of Energy.— 2021.— 51 pp. ↑83, 89, 90, 93, 115
- [57] Harrison R. J. *Performance engineering on A64FX with SVE intrinsics (Early experience on Ookami)*.— 2021.— 37 pp. [URL](#) ↑83, 84, 85, 88, 96, 98
- [58] Michalowicz B., Raut E., Kang Y., Curtis T., Chapman B., Oryspayev D. *Comparing the behavior of OpenMP Implementations with various Applications on two different Fujitsu A64FX platforms*.— 2021. [arXiv](#) [2106.09787](#) ↑115, 116, 117
- [59] Bari M. A. S., Chapman B., Curtis A., Harrison R. J., Siegmann E., N. A. Simakov, M. D. Jones *A64FX performance: experience on Ookami*, 2021 IEEE International Conference on Cluster Computing (CLUSTER) (7–10 Sept. 2021, Portland, OR, USA).— pp. 711–718. [doi](#) ↑84, 85, 86, 87, 88, 90, 96, 102, 103, 104, 116

- [60] Meng J., Atle A., Calandra H., Araya-Polo M. *Minimod: A finite difference solver for seismic modeling.*— 2020. arXiv:2007.06048 ↑^{84, 117}
- [61] Bailey D., Harris T., Saphir W., van der Wijngaart R., Woo A., Yarro M. *The NAS parallel benchmarks 2.0*, Report NAS-95-020: NASA Ames Research Center.— 1995.— 24 pp. URL ↑
- [62] Murai H. *Overview of software environment on Fugaku*, 6th Meeting for Application Code Tuning on A64FX Computer Systems (June 30, 2021).— 17 pp. URL ↑⁸⁷
- [63] Hammond S., Curry M., Davis K., Dang V.-Q., Guba O., Hoekstra R., Laros J., Pedretti K., Poliakoff D., Rajamanickam S., Trott C., Vergiat-Berger L., Younge A. *Fugaku and A64FX Update.*— 2021.— 15 pp. URL ↑^{88, 91, 96}
- [64] Xu R.-Q. G., Okubo T., Todo S., Imada M. *Optimized implementation for calculation and fast-update of Pfaffians installed to the open-source fermionic variational solver mVMC.*— 2021. arXiv:2105.13098 ↑^{89, 111}
- [65] Van Zee F. G., van de Geijn R. A. *BLIS: a framework for rapidly instantiating BLAS functionality* // ACM Transactions on Mathematical Software.— 2015.— Vol. **41**.— No. 3.— pp. 1–33, 14. doi ↑⁸⁹
- [66] Imamura T. *Development of EigenExa from K to Fugaku, and beyond Fugaku*, The 4th Meeting for Application Code Tuning on A64FX Computer Systems (March 17, 2021).— 2021.— 24 pp. URL ↑^{89, 94}
- [67] Shibata N., Petrogalli F. *SLEEF: A portable vectorized library of C standard mathematical functions* // IEEE Transactions on Parallel and Distributed Systems.— 2019.— Vol. **31**.— No. 6.— pp. 1316–1327. doi ↑⁸⁹
- [68] Feldman C., Michalowicz B., Calder A. *Lessons Learned. An In-Depth Look at Running FLASH on Ookami.*— 30 pp. URL ↑⁸⁹
- [69] Ruhela A., Xu S., Manian K. V., Subramoni H., Panda D. K. *Analyzing and understanding the impact of interconnect performance on HPC, Big Data, and deep learning applications: a case study with InfiniBand EDR and HDR*, 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) (18–22 May 2020, New Orleans, LA, USA).— pp. 869–878. doi ↑⁹⁰
- [70] Trott C., Lebrun-Grandié D., Arndt D., Ciesko J., Dang V., Ellingwood N., Gayatri R., Harvey E., Hollman D. S., Ibanez D., Liber N., Madsen J., Miles J., Poliakoff D., Powell A., Rajamanickam S., Simberg M., Sunderland D., Turcsin B., Wilke J. *Kokkos 3: Programming model extensions for the exascale era* // IEEE Transactions on Parallel and Distributed Systems.— 2022.— Vol. **33**.— No. 4.— pp. 805–817. doi ↑⁹¹
- [71] Kudo S., Nitadori K., Ina T., Imamura T. *Implementation and numerical techniques for one EFlop/s HPL-AI benchmark on Fugaku*, 2020 IEEE/ACM 11th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA) (13 Nov. 2020, Atlanta, GA, USA).— pp. 69–76. doi ↑^{92, 117}
- [72] Kudo S., Nitadori K., Ina T., Imamura T. *Prompt report on Exa-scale HPL-AI benchmark*, 2020 IEEE International Conference on Cluster

- Computing (CLUSTER) (14–17 Sept. 2020, Kobe, Japan).– pp. 418–419.  [↑92](#)
- [73] Эйсымонт Л., Фролов А., Семенов А. *Graph500: адекватный рейтинг // Открытые системы. СУБД.*– 2011.– № 7.– с. 14–17.  [↑92](#)
- [74] Nakao M., Ueno K., Fujisawa K., Kodama Y., Sato M. *Performance evaluation of supercomputer Fugaku using breadth-first search benchmark in Graph500*, 2020 IEEE International Conference on Cluster Computing (CLUSTER) (14–17 Sept. 2020, Kobe, Japan).– pp. 408–409.  [↑92](#), 103
- [75] Nakamura Y. *Software development and performance of Fugaku and ARM architectures*, The 38th International Symposium on Lattice Field Theory (26–30 July 2021).– 2021.– 9 pp.  [↑92](#), 113
- [76] Ishikawa K. I., Kanamori I., Matsufuru H., Miyoshi I., Mukai Y., Nakamura Y., Nitadori K., Tsuji M. *102 PFLOPS Lattice QCD quark solver on Fugaku.*– 2021. [arXiv:2109.10687](#)  [↑92](#), 113
- [77] Yashiro H., Koji T., Yuta K., Shuhei K., Takemasa M., Toshiyuki I., Kazuo M., Masuo N., Chihiro K., Masaki S., Hirofumi T. *The NICAM 3.5 km-1024 ensemble simulation: Performance optimization and scalability of NICAM-LETKF on supercomputer Fugaku*, vEGU21, the 23rd EGU General Assembly (online 19–30 April, 2021), EGU21-4771.   [↑93](#)
- [78] Dongarra J., Hammarling S., Higham N. J., Relton S. D., Valero-Lara P., Zounon M. *The design and performance of batched BLAS on modern high-performance computing systems // Procedia Computer Science.*– 2017.– Vol. **108.**– pp. 495–504.  [↑96](#)
- [79] Shimizu T. *Supercomputer Fugaku: Co-designed with application developers/researchers*, 2020 IEEE Asian Solid-State Circuits Conference (A-SSCC) (9–11 Nov. 2020, Hiroshima, Japan).– pp. 1–4.  [↑97](#)
- [80] Deakin T., Price J., Martineau M., McIntosh-Smith S. *GPU-STREAM v2.0: Benchmarking the achievable memory bandwidth of many-core processors across diverse parallel programming models // ISC High Performance 2016: High Performance Computing*, International Conference on High Performance Computing, Lecture Notes in Computer Science.– vol. **9945**, Cham:Springer.– 2016.– ISBN 978-3-319-46079-6.– pp. 489–507.  [↑97](#)
- [81] McVoy L. W., Staelin C. *lmbench: portable tools for performance analysis*, ATEC '96: Proceedings of the 1996 annual conference on USENIX Annual Technical Conference (22–26 January, 1996, San Diego, CA, USA).– 17 pp.  [↑97](#)
- [82] Gupta N., Ashiwal R., Brank B., Peddoju S. K., Pleiter D. *Performance evaluation of parallel execution model on arm-based platforms*, 2020 IEEE International Conference on Cluster Computing (CLUSTER) (14–17 Sept. 2020, Kobe, Japan).– pp. 567–575.  [↑98](#)
- [83] Odajima T., Kodama Y., Tsuji M., Matsuda M., Maruyama Y., Sato M. *Preliminary performance evaluation of the Fujitsu A64FX using HPC applications*, 2020 IEEE International Conference on Cluster Computing (CLUSTER) (14–17 Sept. 2020, Kobe, Japan).– pp. 523–530.  [↑99](#), 102, 109, 110, 116

- [84] Ltaief H., Cranney J., Gratadour D., Hong Y., Gattineau L., Keyes D. E. *Meeting the real-time challenges of ground-based telescopes using low-rank matrix computations.*— 2021.  [URL](#) ^{↑99, 117}
- [85] Кузьминский М. *Векторные процессоры против акселераторов // Открытые системы. СУБД.*— 2018.— № 1.— с. 10–10.   ^{↑99}
- [86] Brank B., Nassyr S., Pouyan F., Pleiter D. *Porting applications to Arm-based processors*, 2020 IEEE International Conference on Cluster Computing (CLUSTER) (14–17 Sept. 2020, Kobe, Japan).— pp. 559–566.  ^{↑100}
- [87] Alappat C., Meyer N., Laukemann J., Gruber T., Hager G., Wellein G., Wettig T. *Execution-Cache-Memory modeling and performance tuning of sparse matrix-vector multiplication and Lattice quantum chromodynamics on A64FX* // *Concurrency and Computation: Practice and Experience*, e6512 (to appear).  ^{↑100, 101, 111, 112}
- [88] Alappat C., Laukemann J., Gruber T., Hager G., Wellein G., Meyer N., Wettig T. *Performance modeling of streaming kernels and sparse matrix-vector multiplication on A64FX*, 2020 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS) (12 Nov., 2020, Atlanta, GA, USA).— pp. 1–7.  ^{↑101}
- [89] Jackson A., Weiland M., Brown N., Turner A., Parsons M. *Investigating applications on the A64FX*, 2020 IEEE International Conference on Cluster Computing (CLUSTER) (14–17 Sept. 2020, Kobe, Japan).— pp. 549–558.  ^{↑101, 102, 108, 115, 116, 118}
- [90] McMahon F. H. *Livermore Fortran kernels: A computer test of numerical performance range*, Technical Report UCRL-53745: Lawrence Livermore National Laboratory.— 1986.— 204 pp. ^{↑102}
- [91] Luszczek P. R., Bailey D. H., Dongarra J. J., Kepner J., Lucas R. F., Rabenseifner R., Takahashi D. *The HPC Challenge (HPCC) benchmark suite*, SC '06: International Conference for High Performance Computing, Networking, Storage and Analysis (11–17 Nov., 2006, Tampa, Florida, USA).— 2006.— pp. 213.  ^{↑99, 102}
- [92] Jin H., Frumkin M., Yan J. *The OpenMP implementation of NAS parallel benchmarks and its performance*, NAS Technical Report NAS-99-011.— 1999.— 26 pp.  ^{↑103}
- [93] Li L., Pandey S., Flynn T., Liu H., Wheeler N., Hoisie A. *SimNet: accurate and high-performance computer architecture simulation using machine learning.*— 2021. arXiv:  2105.05821 ^{↑103}
- [94] Kodama Y., Kondo M., Sato M. *Evaluation of SPEC CPU and SPEC OMP on the A64FX*, 2021 IEEE International Conference on Cluster Computing (CLUSTER) (7–10 Sept. 2021, Portland, OR, USA).— pp. 553–561.  ^{↑103}
- [95] Nakao M., Ueno K., Fujisawa K., Kodama Y., Sato M. *Performance of the supercomputer Fugaku for breadth-first search in Graph500 benchmark // ISC High Performance 2021: High Performance Computing*, International Conference on High performance Computing, Lecture Notes in Computer

- Science.— vol. **12728**, Cham:Springer.— 2021.— ISBN 978-3-030-78713-4.— pp. 372–390. [doi](#) ↑103
- [96] Nakao M. *Performance tuning of Graph500 benchmark on supercomputer Fugaku*, The first Meeting for Application Code Tuning on A64FX Computer Systems (9 December, 2020).— 23 pp. [URL](#) ↑103
- [97] Bramas B. *A fast vectorized sorting implementation based on the ARM scalable vector extension (SVE)*.— 2021. [arXiv](#) [2105.07782](#) ↑104
- [98] Furuya A., Asami A. *Fujitsu regarding OSS porting for Tomitake/FX1000-RIST co-creation* (in Japanese).— 12 pp. [URL](#) ↑106
- [99] Zempo Y., Akino N., Ishida M., Tomiyama E., Yamamoto H. *Real-time and real-space program tuned in K-computer* // Journal of Physics: Conference Series.— 2015.— Vol. **640**.— No. 1, 012066. [doi](#) ↑109
- [100] Nakajima T., Katouda M., Kamiya M., Nakatsuka Y. *NTChem: A high-performance software package for quantum molecular simulation* // International Journal of Quantum Chemistry.— 2015.— Vol. **115**.— No. 5.— pp. 349–359. [doi](#) ↑109
- [101] Sato M., Murai H., Nakao M., Tsugane K., Odajima T., Lee J. *XcalableMP 2.0 and future directions* // *XcalableMP PGAS Programming Language*, ed. M. Sato, Singapore:Springer.— 2021.— ISBN 978-981-15-7683-6.— pp. 245–262. [doi](#) ↑109
- [102] Poenaru A., Lin W. C., McIntosh-Smith S. *A performance analysis of modern parallel programming models using a compute-bound application* // *ISC High Performance 2021: High Performance Computing*, International Conference on High Performance Computing, Lecture Notes in Computer Science.— vol. **12728**, Cham:Springer.— 2021.— ISBN 978-3-030-78713-4.— pp. 332–350. [doi](#) ↑110
- [103] Watanabe K. *Performance tuning on LAMMPS for A64FX system*, The 5th Meeting for Application Code Tuning on A64FX Computer Systems (27 April, 2021).— 2021.— 27 pp. [URL](#) ↑110
- [104] Huber J., Wei W., Georgakoudis G., Doerfert J., Hernandez O. *A case study of LLVM-based analysis for optimizing SIMD code generation* // *IWOMP 2021: OpenMP: Enabling Massive Node-Level Parallelism*, International Workshop on OpenMP, Lecture Notes in Computer Science.— vol. **12870**, Cham:Springer.— 2021.— ISBN 978-3-030-85262-7.— pp. 142–155. [doi](#) ↑111
- [105] Kanamori I., Ishikawa K. I., Matsufuru H. *Object-oriented implementation of algebraic multi-grid solver for lattice QCD on SIMD architectures and GPU clusters* // *ICCSA 2021: Computational Science and Its Applications*, International Conference on Computational Science and Its Applications, Lecture Notes in Computer Science.— vol. **12953**, Cham:Springer.— 2021.— ISBN 978-3-030-86976-2.— pp. 218–233. [doi](#) ↑111, 113
- [106] Nakamura Y., Mukai Y., Ishikawa K.-I., Kanamori I., Lattice quantum chromodynamics simulation library for Fugaku and computers with wide SIMD. [URL](#) ↑113

- [107] Cielo S., Porth O., Iapichino L., Karmakar A., Olivares H., Xia C. *Optimizing the hybrid parallelization of BHAC*.– 2021. arXiv:2108.12240 ↑¹¹³
- [108] Bird R., Tan N., Luedtke S. V., Harrell S.-L., Taufer M., Albright B. *VPIC 2.0: next generation particle-in-cell simulations*.– 2021. arXiv:2102.13133 ↑¹¹⁴
- [109] Klöwer M., Hatfield S., Croci M., Düben P. D., Palme T. N. *Fluid simulations accelerated with 16 bit: Approaching 4x speedup on A64FX by squeezing ShallowWaters.jl into Float16*: ESSOAr.– 2021.– 26 pp. doi URL ↑¹¹⁶
- [110] Ferenbaugh C. R. *PENNANT: an unstructured mesh mini-app for advanced architecture research* // Concurrency and Computation: Practice and Experience.– 2015.– Vol. 27.– No. 17.– pp. 4555–4572. doi ↑¹¹⁶
- [111] Nukariya A., Akao K., Takahashi J., Fukumoto N., Kawakami K., Kuroda A., Minami K., Sato K., Matsuoka S. *HPC and AI initiatives for supercomputer Fugaku and future prospects*, Fujitsu Technical Review.– 6 pp. URL ↑¹¹⁸
- [112] Rajpal S., Lakhiani N., Singh A.-K., Kohli R., Kumar N. *Using handpicked features in conjunction with ResNet-50 for improved detection of COVID-19 from chest X-ray images* // Chaos, Solitons & Fractals.– 2021.– Vol. 145, 110749. doi ↑¹¹⁸
- [113] Honda T. *Development of a deep neural network library for A64FX*, R-CCS 4th Meeting for Application Code Tuning on A64FX Computer Systems (17 March 2021).– 25 pp. URL ↑¹¹⁸

Поступила в редакцию 17.12.2021;
 одобрена после рецензирования 21.12.2021;
 принята к публикации 22.02.2022.

Рекомендовал к публикации

д.ф.-м.н. С. М. Абрамов

Информация об авторе:



Михаил Борисович Кузьминский

старший научный сотрудник лаборатории компьютерного обеспечения химических исследований, кандидат химических наук ИОХ РАН. Научные интересы — высокопроизводительные вычисления, аппаратура ЭВМ, вычислительная химия.



0000-0002-3944-8203

e-mail: kus@free.net