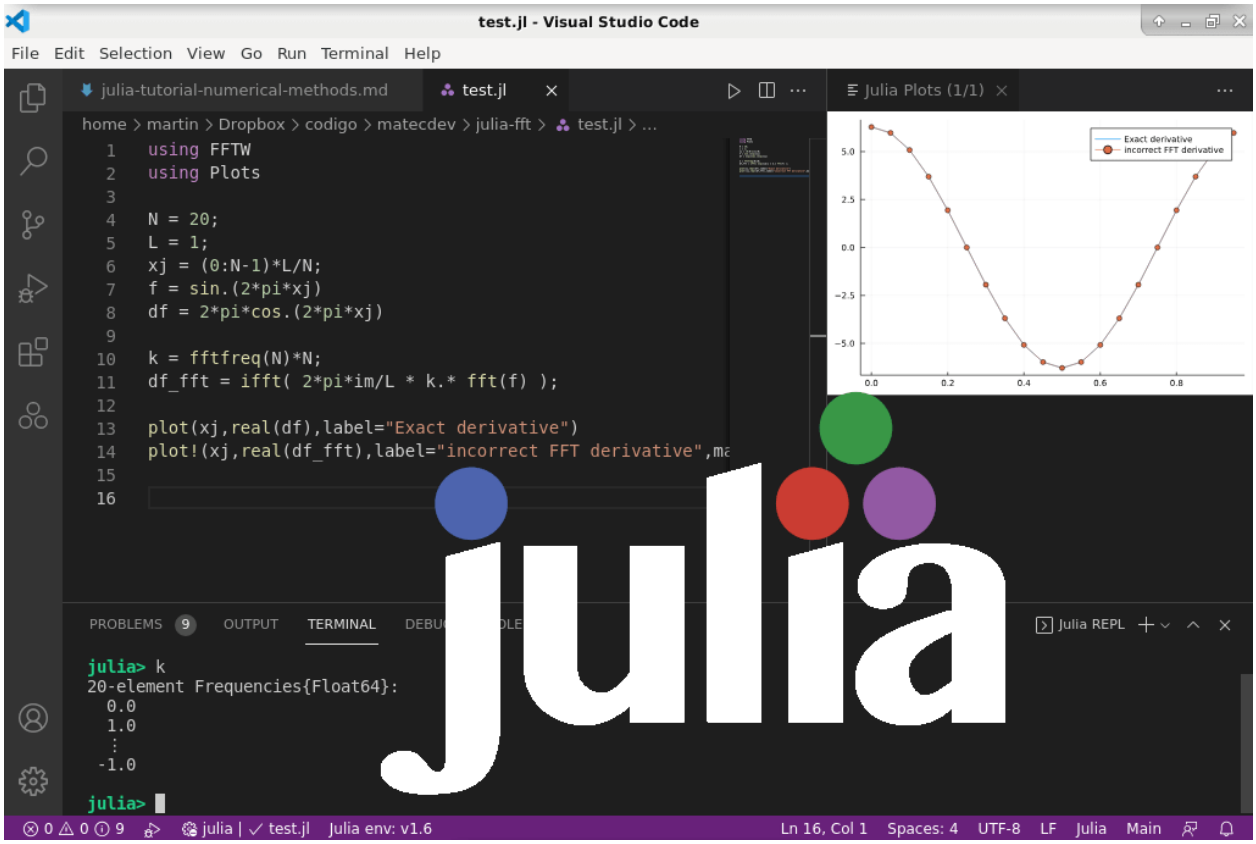
1.1 JULIA новый язык программирования для научных расчетов

Когда речь заходит о новом языке программирования, для пользователя это всегда значит необходимость по-новому формулировать свои мысли и идеи. Переход на новый язык будет оправдан, если он не слишком сложен для освоения, а преимущества его использования заметны. Язык Julia, который появился относительно недавно, — это открытый динамический язык, созданный в первую очередь для высокопроизводительных вычислений в научных и технических областях.



Раньше казалось, что интерактивность — это прерогатива только интерпретируемых языков, таких как Python, R или MATLAB. Но Julia меняет правила игры: благодаря JIT-компиляции (которая срабатывает только тогда, когда код нужно выполнить) и множественной диспетчеризации (функции вызываются в зависимости от типов переданных параметров), она предлагает почти такую же интерактивность, при этом скорость выполнения программ сравнима со скоростью C.

В Julia есть практически все, что можно найти в популярных языках: различные числовые типы, включая числа с произвольной точностью, богатый набор математических функций, современные структуры данных (кортежи, словари, множества), возможность интроспекции кода, встроенный менеджер пакетов и взаимодействие с другими языками и библиотеками. Но есть и новшества: высокоуровневые инструменты для параллельных и

распределенных вычислений, поддержка метапрограммирования и возможность использовать широкий диапазон символов в коде. Это может создавать некоторые трудности при написании и представлении Julia-кода в статьях или книгах.

Главное, на что хочется обратить внимание: язык Julia — это универсальный и мощный инструмент, который может быть полезен каждому исследователю в любой области науки или техники. Язык программирования Julia становится сильной альтернативой Python и C++ для рабочих процессов анализа данных в области физики высоких энергий. В нашем исследовании мы сравниваем производительность различных реализаций на Julia с теми, что на Python и C++, показывая, что Julia превосходит в задачах, где основным фактором является вычислительная нагрузка, в то время как для задач, доминируемых доступом к данным, мы демонстрируем решение, включающее параллельное чтение файлов и обработку данных.

В области байесовского вывода продвинутые алгоритмы необходимы для эффективного анализа данных. Набор инструментов байесовского анализа (BAT) служит комплексной коллекцией алгоритмов и методов, предназначенных для облегчения применения байесовской статистики к сложным пользовательским задачам. Введение BAT.jl знаменует собой современную переимплементацию BAT на Julia, использующую модульный дизайн программного обеспечения, который поддерживает параллельные и распределенные вычисления. Это улучшение включает новые алгоритмы выборки и интеграции, что позиционирует BAT.jl как высокопроизводительную платформу для байесовского вывода, соответствующую требованиям современного анализа данных.

Байесовский вывод часто представляет собой вычислительные задачи, требующие эффективных алгоритмов и инструментов для решения сложных проблем. BAT использует методы марковских цепей Монте-Карло для предоставления надежного инструментария для решения байесовских задач. Для повышения своей эффективности и применимости в настоящее время проводится полная переработка BAT с использованием Julia в качестве основного языка разработки и сосредоточением на инновационных алгоритмах выборки для улучшения производительности и сокращения времени вычислений. Эта переработка направлена на минимизацию зависимостей программного обеспечения и создание интерфейсов с широко используемыми языками и инструментами, что расширяет применимость BAT в различных областях исследований.

Важность статистических методов в научных исследованиях трудно переоценить. Эта диссертация состоит из двух основных компонентов: разработки алгоритмов для оценки байесовского доказательства и параллелизации методов марковских цепей Монте-

Карло. Кроме того, она включает тщательный анализ параметров пучков протонов в эксперименте AWAKE в CERN, используя передовые статистические методы.

Использование JULIA в расчетных исследованиях в физике высоких энергий (HEP)

Требования к вычислительной технике HEP

Поскольку программные коды, используемые в исследованиях HEP, очень большие, с высокой степенью взаимозависимости, в коде обычно используется множество библиотек с открытым исходным кодом, разработанных другими авторами; таким образом, усилия по смене языка программирования являются значительными. Переход на новый язык может произойти только в том случае, если он дает существенное преимущество по сравнению с уже используемой парадигмой. Ключевое преимущество Julia, которое может сделать переход на новый язык целесообразным, - это упрощение, которое возникнет при использовании одного языка вместо комбинации двух, C++ и Python. Вычисления в HEP очень обширны и включают в себя множество вариантов использования: автоматизация управления экспериментом, сбор данных, генерация феноменологических и физических событий, моделирование физического эксперимента, реконструкция физических событий из записанных данных, анализ реконструированных событий и многое другое.

Общие свойства

А) Простой язык

Простота языка - это одна сторона свойства высокоуровневого и высокопроизводительного языка, которое послужило бы мотивом для принятия Julia в качестве языка программирования. Он прост, по крайней мере, в двух отношениях: простой, императивный синтаксис и отсутствие сильной типизации при написании кода. Поверхностный синтаксис Julia во многом напоминает Python, MATLAB (поток управления, литеральный массив), а также черпает вдохновение, например, блок do, из Lua и Ruby. В нем есть все, что можно ожидать от языка высокого уровня: функции высшего порядка (функции можно возвращать и передавать как переменные), собственные N-мерные массивы, вложенные нерегулярные массивы (массивы из массивов разного размера) и синтаксис для трансляции по массивам.

Б) Производительность

Еще одно важное преимущество Julia - высокая производительность. Julia обеспечивает производительность, схожую с C++, а в некоторых случаях даже превосходящую C++, как видно на рис. 3. Показанное сравнение получено путем

повторения микробенчмарка5 для Julia версии 1.9.0rc1 и Python 3.9.2. Реализации на C и Julia используют OpenBLAS для матричных операций, а Python использует NumPy (версия 1.24.2) и OpenBLAS. В этом бенчмарке сравнивается время выполнения некоторых коротких алгоритмов, реализованных схожим образом на разных языках. Результаты делятся на время, затраченное на реализацию на языке C/C++. Использовался компилятор GNU gcc версии 10.2.1. Тест выполнен на ноутбуке, оснащенном процессором 11-го поколения Intel(R) Core(TM) i5-1135G7@2.40GHz и 16 ГБ оперативной памяти (RAM) под управлением операционной системы Linux, скомпилированной под 64-битную архитектуру x86. Используется библиотека OpenBLAS 64-битной версии 0.3.21. Данная конфигурация используется для всех тестов производительности, описанных в данной статье, если не указано иное. Оценка колеблется от 0,73 до 1,67 (меньше - лучше) для Julia и от 1,12 до 107 для Python. Си показывает наилучшие результаты по сравнению с двумя другими языками для рекурсивного алгоритма Фибоначчи..

B) Взаимодействие с унаследованным кодом

Вычисления на HEP основаны на наследии программного кода, написанного в течение десятилетий. Взаимодействие с библиотеками, разработанными на C++ и Fortran, неизбежно, за исключением последнего шага анализа области (и даже здесь это было бы привлекательной особенностью). Julia может нативно вызывать функции C и Fortran без каких-либо накладных расходов по сравнению с вызовом их из родного языка.

Выводы

Язык программирования Julia был представлен и сравнен с C++ и Python. Для изучения потенциала Julia для HEP был составлен список требований к автономным и программно-триггерным приложениям HEP, охватывающий как сам язык, так и его экосистему. Была изучена совместимость Julia с этими требованиями. Julia и ее экосистема впечатляюще удовлетворяют всем этим требованиям. Более того, Julia предлагает другие возможности - интегрированную систему упаковки с поддержкой воспроизводимости, множественную отправку и автоматическую дифференциацию, - от которых HEP-приложения только выиграют. Способность обеспечивать одновременно простоту программирования и производительность делает Julia идеальным языком программирования для анализа данных HEP и в целом является важным преимуществом для всех рассматриваемых приложений HEP. Динамическая парадигма множественной диспетчеризации, используемая в Julia, как оказалось, облегчает повторное использование кода. Это свойство принесет большую пользу приложениям сообщества HEP, в разработке которых участвует множество людей из разных групп. Использование единого и простого

языка программирования облегчит обучение. Опыт показывает, что студенты, владеющие C++ или Python, очень быстро осваивают язык и становятся продуктивными уже через несколько дней. Использование Julia в качестве основного языка в совместной работе позволяет студентам, работающим над краткосрочными проектами, использовать общий язык программирования, в то время как в случае с C++ часто требуется использование более простого языка, например Python. Это облегчает повторное использование кода, разработанного в таком контексте. Мы измерили производительность языка в контексте анализа данных HEP. Измерения показали отличную производительность во время выполнения, конкурентоспособную с C++: на 11 % быстрее для простого примера анализа событий LHC, использованного в качестве эталона. По сравнению с Python, язык не только быстрее, но и гораздо менее чувствителен к выбору реализации. Было показано, что реализация на Python на три порядка медленнее, чем Julia, когда цикл обработки событий выполняется на Python. Векторизация может быть использована для перемещения цикла событий с помощью базовых компилируемых библиотек, и это сокращает разрыв в производительности. Разница между C++ и Python заключается в том, что Julia моложе и имеет меньшее сообщество. Сообщество Julia очень активно сотрудничает, и, несмотря на его меньшую популярность, информацию о разработке на этом языке легко найти в Интернете. Быстрый рост Julia в научных кругах и промышленности дает нам уверенность в долгосрочной преемственности языка Julia, что очень важно для проектов HEP, поскольку они имеют большой временной диапазон. Учитывая результаты данного исследования, сообщество HEP, безусловно, выиграет от широкомасштабного внедрения языка программирования Julia для разработки программного обеспечения. Консолидация базовых библиотек, специфичных для HEP, будет иметь большое значение для облегчения этого процесса.

Таким образом, в конце можно выделить следующие основные тенденции:

1. Принятие Julia: Наблюдается растущее признание Julia как мощного инструмента для анализа данных в области физики высоких энергий, особенно для вычислительно интенсивных задач.
2. Фокус на байесовских методах: Акцент на байесовском анализе подчеркивает необходимость в сложных алгоритмах и инструментах для эффективного решения сложных задач с данными.
3. Улучшения производительности: Переосмысление BAT на Julia направлено на повышение производительности через инновационные алгоритмы и улучшение

удобства использования, что отражает текущую тенденцию к оптимизации вычислительной эффективности в анализе данных.

4. Междисциплинарные приложения: Упор на минимизацию зависимостей и создание интерфейсов с другими инструментами свидетельствует о тенденции сделать байесовский анализ доступным в различных областях исследований.

Источники:

- Stanitzki, Marcel, & Strube, Jan (2020). Performance of Julia for high energy physics analyses (DESY--20-056). Germany
- Caldwell, Allen, Grunwald, Cornelius, Hafych, Vasyl, Kröninger, Kevin, & Cagnina, Salvatore La (2020). BATjl Upgrading the Bayesian Analysis Toolkit. EPJ Web of Conferences, v. doi:101051/epjconf/202024506001
- Grunwald, Cornelius, Kröninger, Kevin, & La Cagnina, Salvatore (2019). BATjl - A new toolkit for Bayesian analysis. Verhandlungen der Deutschen Physikalischen Gesellschaft, (Aache2019issue), 1.
- Hafych, Vasyl (Jan 2022). Development of advanced statistical algorithms and application to the AWAKE experiment at CERN. Available from INIS: http://inis.iaea.org/search/search.aspx?orig_q=RN:53109344; Available from INIS in electronic form. Also available from: http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20220203-1632505-1-7