УДК 004.8:364

Широков Руслан Эдуардович, магистрант, ИСОиП (филиал) ДГТУ в г. Шахты, г. Шахты

АРХИТЕКТУРА ПРОГРАММНОГО РЕШЕНИЯ ДЛЯ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ СОЦИАЛЬНОГО ОБЕСПЕЧЕНИЯ НА ОСНОВЕ PYTHON И C++

Аннотация

В статье рассматривается архитектура программного решения для интеллектуального анализа данных социального обеспечения на основе языков Python и C++. Актуальность темы связана с ростом объема данных, которые используются при назначении мер социальной поддержки, контроле выплат и оценке бюджетной нагрузки. Отдельное внимание уделено проблемам интеграции разнородных источников, обработке больших массивов сведений и соблюдению требований к защите персональных данных. В работе предложена многоуровневая структура системы, включающая интеграционный уровень, уровень хранения и подготовки данных, аналитическое ядро, сервисный уровень, а также уровень безопасности и аудита. Python рассматривается как инструмент для подготовки данных, построения аналитических сценариев, прототипирования моделей и разработки сервисной логики. C++ предлагается использовать для ресурсоемких модулей, где требуется высокая скорость обработки, контроль памяти и эффективная работа с большими массивами записей. В статье описана роль pybind11 при связывании Python-кода с вычислительными модулями на C++. Также раскрывается значение Apache Arrow как формата, который снижает количество лишних преобразований данных при передаче между компонентами системы. Предложенный подход позволяет совместить

гибкость аналитической разработки с высокой производительностью вычислительного ядра. Материалы статьи могут быть использованы при проектировании информационных систем для социальной сферы, где одновременно требуются точность обработки данных, масштабируемость и контроль доступа к чувствительной информации.

Annotation

The article examines the architecture of a software solution for intelligent analysis of social security data based on Python and C++. The relevance of the topic is connected with the growing volume of data used for assigning social support measures, monitoring payments, and estimating budgetary load. Special attention is paid to the problems of integrating heterogeneous sources, processing large datasets, and complying with personal data protection requirements. The paper proposes a multi-level system structure that includes an integration layer, a data storage and preparation layer, an analytical core, a service layer, and a security and audit layer. Python is considered as a tool for data preparation, development of analytical scenarios, model prototyping, and service logic implementation. C++ is proposed for resource-intensive modules that require high processing speed, memory control, and efficient work with large arrays of records. The article describes the role of pybind11 in connecting Python code with computational modules written in C++. The importance of Apache Arrow is also discussed as a format that reduces unnecessary data transformations when information is transferred between system components. The proposed approach makes it possible to combine the flexibility of analytical development with the high performance of the computational core. The materials of the article can be used in the design of information systems for the social sphere, where data accuracy, scalability, and controlled access to sensitive information are required at the same time.

Ключевые слова: интеллектуальный анализ данных, социальное обеспечение, программная архитектура, Python, C++, персональные данные, аналитическое ядро, Apache Arrow.
Keywords: intelligent data analysis, social security, software architecture, Python, C++, personal data, analytical core, Apache Arrow.

Цифровизация социальной сферы привела к резкому росту объема данных, которые используются при назначении, учете и контроле мер социальной поддержки. Для повышения эффективности принятия управленческих решений была создана Единая государственная информационная система социального обеспечения (ЕГИССО). Она аккумулирует информацию о мерах социальной защиты, социальных гарантиях и категориях получателей, что позволяет не только вести учет, но и эффективно проводить аналитические расчеты для принятия управленческих решений [1]. В этих условиях программное решение должно решать три задачи одновременно: интегрировать разнородные источники данных, быстро обрабатывать большие массивы информации и соблюдать требования к защите персональных данных, что особенно важно в контексте работы с чувствительной информацией, такой как данные о социальном обеспечении [9].

Интеллектуальный анализ данных в государственном секторе помогает выявлять аномалии, повышать точность управленческих решений, контролировать бюджетные расходы и предотвращать ошибки, такие как дублирование данных или необоснованные назначения выплат [2, с. 23–25]. Для социальной сферы это особенно важно, потому что ошибка в данных влияет уже не на абстрактный показатель, а на назначение выплаты, подтверждение нуждаемости или расчет бюджетной нагрузки. При этом архитектура таких систем усложняется из-за фрагментированности

источников, различий в форматах сведений и ограничений на обработку персональных данных [3, с. 3].

По этой причине для системы, которая будет интеллектуально анализировать данные социального страхования, логично использовать многоуровневую архитектуру. Первый уровень архитектуры — это «интеграционный уровень». Интеграционный уровень обеспечивает связь между всеми различными департаментами и муниципалитетами в их соответствующих информационных системах, проверяет форматы предоставления данных, наличие обязательных полей и идентификаторов, а также проверяет согласованность записей. Второй уровень называется «хранение и подготовка данных». Этот уровень обеспечивает операционную структуру для хранения и управления данными, включая основные записи и аналитические витрины, которые будут использоваться для моделирования, составления отчетов и выявления аномалий. Третий уровень, называемый «аналитическим ядром», отвечает за дедупликацию получателей, сегментацию получателей, выявление аномалий, оценку риска неправильного распределения и прогнозирование расходов. Четвертый уровень — это «сервисный уровень», где мы предоставляем пользователям доступ к интерфейсам, внутренним API, панелям мониторинга, картам рисков или формам отчетности. И наконец, на «уровне безопасности и аудита» мы контролируем, кто может что делать, когда, и регистрируем каждое действие, совершаемое каждым пользователем, а также устанавливаем ограничения на скачивание [4, с. 2; 9].

Использование Python и C++ в такой структуре напрямую отражает характер обрабатываемых задач. Python — отличный язык для тех областей системы, которые требуют быстрого изменения или разработки новых аналитических ситуаций. Он хорошо подходит для разработки аналитических решений, требующих быстрого прототипирования, включая предварительную обработку данных (очистка, объединение таблиц,

создание признаков), статистический анализ, моделирование и построение сервисного слоя. Важно также наличие развитой экосистемы аналитических библиотек. Например, библиотека Pandas в своей документации позиционирует себя как «быстрый и гибкий способ анализа и обработки данных» [5]. Это важно для социального сектора, поскольку данные о пособиях, платежах и получателях обычно содержат пропущенные значения, повторяющиеся значения, несоответствия в коде и различные форматы данных.

Тем не менее, Python не подходит для всех задач промышленной аналитики. Когда обрабатываются миллионы записей одновременно, накладные расходы, такие как сериализация, передача данных между процессами и использование памяти, могут стать огромной нагрузкой. В документации Python модуль multiprocessing.shared_memory используется как средство для доступа нескольких процессов к одному и тому же месту в памяти [6]. Это типичная проблема анализа нагрузки; по сути, если вы постоянно копируете массивы данных между компонентами системы, вы фактически сделаете свою систему неработоспособной. Это особенно актуально в приложениях социального обеспечения для обнаружения дубликатов, проверки пакетного назначения, оценки рисков и расчета множества аналитических показателей в реальном времени.

Обоснованием разработки ресурсоемких модулей на языке программирования C++ является то, что разработчик получает больший контроль над управлением памятью компьютера, улучшенную многопоточную обработку кода приложения и возможность ускорять критически важные участки системы без необходимости переписывать остальную часть системы. К различным типам модулей, которые можно разрабатывать на C++, относятся: сопоставление записей, обнаружение аномалий, индексирование больших массивов, пакетное применение моделей и обработка потоков данных.

Сочетание Python и C++ дает множество практических преимуществ, поскольку систему можно разделить на функциональные секции. Python служит языком программирования для слоя оркестровки и аналитики, а C++ — для вычислительного ядра. Для их соединения можно использовать pybind11; pybind11 — это библиотека для разработки привязок Python к коду C++. Этот подход предоставляет модулю аналитики возможность использовать нативные компоненты в качестве типичных функций и в конечном итоге не меняет архитектуру приложения.

Эффективность этого метода можно дополнительно повысить, если передача данных между различными элементами будет основана на столбцовых форматах памяти. Формат Apache Arrow — это языко-нейтральный формат, разработанный для обеспечения очень быстрой аналитики и практически полного отсутствия обмена копиями [8]. Таким образом, при рассмотрении гибридной архитектуры, меньшее количество ненужных преобразований данных и копий массивов приведет к большей отказоустойчивости системы при ее использовании.

Сочетание Python и C++ дает два основных преимущества: во-первых, поскольку код на Python можно писать гораздо быстрее для таких вещей, как аналитические скрипты, хранилища данных и т. д., разработка вашего проекта на Python займет гораздо меньше времени, чем разработка всего проекта на C++. Во-вторых, общая производительность ваших критически важных модулей останется очень хорошей, поскольку вы переносите сложные вычисления в C++. Если бы вы создали всю свою систему на Python, вы бы обнаружили, что при работе с большими наборами данных ваша система либо работала бы хуже, либо, возможно, вообще не работала бы. Если бы вы создали всю свою систему на C++, вы бы обнаружили, что разработка и поддержка аналитических сценариев в постоянно меняющейся среде, например, в социальной сфере, где правила назначения мер поддержки, состав данных и требования к участию

постоянно меняются, займут значительно больше времени (более высокие затраты). В таких ситуациях гибридная архитектура, подобная этой паре технологий, будет особенно полезна.

Еще один аргумент связан с требованиями защиты персональных данных. Федеральный закон 152-ФЗ гласит, что права граждан должны соблюдаться при обработке персональных данных. Поэтому первоначальный проект должен включать следующее: ролевой доступ и аудит транзакций, а также контроль количества загрузок и копий конфиденциальных данных. Разделение уровней в гибридной архитектуре упрощает построение такой организации; например, циклы интеграции, анализа и обслуживания могут быть построены с различными уровнями доступа и различной степенью детализации данных. Таким образом, с точки зрения интеллектуального анализа данных социального страхования, наилучшим вариантом будет многоуровневая архитектура, где Python будет использоваться для подготовки данных, анализа и интеграции с внешними сервисами, а C++ — для вычислительного ядра и модулей ускорения. Такой подход лучше соответствует реальным условиям в государственном секторе, а именно: очень большим разнородным массивам данных, быстрым изменениям аналитической логики, высоким уровням производительности и информационной безопасности.

Список литературы

1. Единая государственная информационная система социального обеспечения [Электронный ресурс] // Социальный фонд России. URL: https://sfr.gov.ru/branches/kurgan/news~2017/06/27/138108

2. Abd Rahman M. S., Hashim H. A., Salleh Z. Data Mining Technology: An Opportunity for Public Sector Accounting // IPN Journal. 2012. Vol. 1, No. 1. P. 13–30. DOI: 10.58458/ipnj.v01.03.02.0016.

3. Voskob M., Punin N. Data mining and Privacy in Public Sector using Intelligent Agents (discussion paper) [Электронный ресурс]. 2003. URL: https://arxiv.org/abs/cs/0311050

4. Avci C., Tekinerdogan B., Athanasiadis I. N. Software architectures for big data: a systematic literature review // Big Data Analytics. 2020. Vol. 5. Art. 16. DOI: 10.1186/s41044-020-00040-7.

5. pandas: Python Data Analysis Library [Электронный ресурс]. URL: https://pandas.pydata.org/

6. multiprocessing.shared_memory — Shared memory for direct access across processes [Электронный ресурс] // Python 3.14.4 documentation. URL: https://docs.python.org/3/library/multiprocessing.shared_memory.html

7. pybind11 documentation [Электронный ресурс]. URL: https://pybind11.readthedocs.io/en/stable/

8. Apache Arrow [Электронный ресурс]. URL: https://arrow.apache.org/

9. О персональных данных: Федеральный закон от 27.07.2006 № 152-ФЗ [Электронный ресурс]. Доступ из справ.-правовой системы / офиц. опубликование правовых актов. URL: https://pravo.gov.ru/

Literature

1. Unified State Information System of Social Security [Electronic resource] // Social Fund of Russia. URL: https://sfr.gov.ru/branches/kurgan/news~2017/06/27/138108

2. Abd Rahman M. S., Hashim H. A., Salleh Z. Data Mining Technology: An Opportunity for Public Sector Accounting // IPN Journal. 2012. Vol. 1, No. 1. P. 13–30. DOI: 10.58458/ipnj.v01.03.02.0016.

3. Voskob M., Punin N. Data Mining and Privacy in Public Sector Using Intelligent Agents (discussion paper) [Electronic resource]. 2003. URL: https://arxiv.org/abs/cs/0311050

4. Avci C., Tekinerdogan B., Athanasiadis I. N. Software Architectures for Big Data: A Systematic Literature Review // Big Data Analytics. 2020. Vol. 5. Art. 16. DOI: 10.1186/s41044-020-00040-7.

5. pandas: Python Data Analysis Library [Electronic resource]. URL: https://pandas.pydata.org/

6. multiprocessing.shared_memory — Shared Memory for Direct Access Across Processes [Electronic resource] // Python 3.14.4 Documentation. URL: https://docs.python.org/3/library/multiprocessing.shared_memory.html

7. pybind11 Documentation [Electronic resource]. URL: https://pybind11.readthedocs.io/en/stable/

8. Apache Arrow [Electronic resource]. URL: https://arrow.apache.org/

9. On Personal Data: Federal Law No. 152-FZ of July 27, 2006 [Electronic resource]. Access from the legal reference system / official publication of legal acts. URL: https://pravo.gov.ru/