PANORAMA DE PYTHON DE ALTO RENDIMIENTO PARA DESARROLLAR DATOS APLICACIONES DE CIENCIA Y APRENDIZAJE AUTOMÁTICO

Kely Zulema Ponce Quispe

17 junio del 2024

Python se utiliza intensamente en los crecientes dominios de la ciencia de datos (DS), la computación científica, análisis de datos y aprendizaje automático (ML). Se utiliza como sucesor de muchos centrados en datos. y lenguajes de programación de computación científica, como R, Fortran y Matlab. Uno de los Las principales razones detrás de este éxito en DS son sus numerosas bibliotecas centradas en DS y ML,

1 INTRODUCCION

Python se utiliza intensamente en los crecientes dominios de la ciencia de datos (DS), la computación científica, análisis de datos y aprendizaje automático (ML). Se utiliza como sucesor de muchos centrados en datos. y lenguajes de programación de computación científica, como R, Fortran y Matlab. Uno de los Las principales razones detrás de este éxito en DS se encuentran en sus numerosas bibliotecas centradas en DS y ML.

2 METODO

Este artículo presenta una revisión narrativa que cubre el dominio de la aceleración del programa Python. ejecución. Por tanto, su objetivo es proporcionar a los profesionales una visión general del estado actual del arte. en programación Python de alto rendimiento, especialmente a través de paralelización, ejecución distribuida, y transformación de código.

Enfoque y contexto

Como con cualquier revisión narrativa, haremos una evaluación cualitativa de los diversos enfoques existentes. en el dominio de la mejora del rendimiento de los programas Python. Para liderar esta encuesta, los autores Confió en su

experiencia, en los campos relacionados, incluida la experiencia industrial y las comunidades de investigación: ML, DS e ingeniería de software.

• El algoritmo desarrollado puede implicar alguna estructura de datos no estándar, como una estructura de datos especial. tipo de gráfico de conocimiento

• Lo más común es que los profesionales de DS se enfrenten a situaciones cercanas a los problemas canónicos que involucran estructuras de datos estándar, como matrices numéricas o gráficos.

• Finalmente, es posible que el algoritmo todavía esté solo en papel y, en lugar de comenzar a implementarlo audazmente en Python vainilla, **Estrategia de búsqueda** Como se indicó anteriormente, nos enfocamos en la mejora del desempeño en el contexto de tres identificados perfiles asociados con los escenarios definidos anteriormente. Para el escenario de la sección, en la Sección 4, motivamos el subconjunto de bibliotecas bajo nuestro enfoque y describimos cómo fue la estrategia de búsqueda. modificado específicamente en ese caso. **Los criterios de inclusión y exclusión** Las publicaciones y herramientas seleccionadas deben proponer alguna mejora del rendimiento de la ejecución. de programas Python que impactan directamente en las tareas de DS o ML. Algunas herramientas con un alcance más amplio pueden incluirse, siempre que permitan claramente acelerar la ejecución de tareas de DS o ML, o función relevante en paquetes de nivel DS o ML.

3 MEJORA DEL RENDIMIENTO DE PYTHON PURO

En esta sección, nos centramos en herramientas y enfoques que soportan la aceleración del código en el que el Las partes computacionalmente intensivas dependen únicamente de la distribución predeterminada de Python. **Enfoques de memoria distribuida y memoria compartida** MPI trabaja con un modelo de memoria distribuida, explotando potencialmente una red distribuida de máquinas. MPI es un estándar de paso de mensajes que define la sintaxis y la semántica de las rutinas de la biblioteca. para desarrollar aplicaciones paralelas. Con MPI, las computadoras que ejecutan un programa paralelo pueden intercambiar mensajes. MPI fue diseñado originalmente para desarrollar programas en los lenguajes C, C++ y Fortran. **Enfoques basados en tareas** Los marcos basados en tareas son herramientas más recientes para paralelizar y distribuir computación pesada. Aunque generalmente son potentes, su uso puede implicar una intensa refactorización del código. Mas complejo Los marcos que imponen un enfoque de desarrollo específico se detallan en la Sección **Transformación y compilación de programas.** Además de las anotaciones, directivas y decoradores para paralelización mencionados en secciones anteriores, La transformación y compilación de programas es otra forma sencilla de obtener un mejor rendimiento de ejecución para una base de código existente con una mínima sobrecarga de trabajo para el profesional.

4 ACELERAR EL USO DE BIBLIOTECAS NUMÉRICAS

En este escenario, el científico de datos ya habría implementado un algoritmo, pero En contraste con la sección anterior, no se basaría solo en Python básico y también Utilice bibliotecas numéricas de Python. De hecho, se habría reconocido que el problema depende principalmente en estructuras de datos estándar, como matrices flotantes, con el objetivo de beneficiarse de las funciones asociadas. primitivas listas para usar **NumPy** NumPy es una de las bibliotecas de Python más utilizadas, ya que proporciona un formato de matriz multidimensional. central para muchas otras bibliotecas. También incluye un conjunto de rutinas para manipular matrices con diferentes operaciones, como primitivas matemáticas, manipulación de formas y clasificación **pandas** Pandas es una biblioteca de Python muy popular para análisis y manipulación de datos. Su formato de marco de datos. se utiliza ampliamente en DS, ya que permite manejar datos heterogéneos, series temporales y manipulación basada en consultas, por nombrar algunas características. **Aprendizaje científico** reutiliza el formato de matriz definido por NumPy pero apunta a una cobertura más completa de conceptos matemáticos y estadísticos de propósito general, como álgebra lineal, pruebas estadísticas, y procesamiento de señales e imágenes. Scikit-learn se basa en NumPy y Scipy mediante la implementación Muchos modelos de la literatura de ML, como modelos de regresión, clasificación y agrupamiento.

5 MARCOS ESTRUCTURANTES

En esta sección, consideramos bibliotecas y marcos de alto rendimiento que imponen un forma de pensar y programar para el practicante y, por lo tanto, se usan preferiblemente justo cuando comienza la implementación. **Marcos de aprendizaje profundo** Muchos modelos utilizados en DS se pueden formalizar como gráficos acíclicos dirigidos (DAG) (por ejemplo, bayesianos).Redes, modelos de mezcla probabilística y, más notablemente, redes neuronales). Una gama de Python Las bibliotecas, comúnmente conocidas como marcos de aprendizaje profundo, vienen con soporte especializado **Marcos de computación distribuida** Un enfoque utilizado por múltiples bibliotecas de computación con uso intensivo de Python es la paralelización basada en tareas, especialmente cuando se trata de grandes conjuntos de datos. El enfoque basado en tareas se refiere a una estrategia. donde el trabajo se divide en múltiples tareas, y estas tareas son manejadas por un administrador de tareas que los asigna a hilos que los ejecutan.

6 DISCUSIÓN Y RESULTADOS

En este artículo, hemos presentado numerosas herramientas y técnicas que proponen mejorar la Rendimiento de Python en el contexto de DS y ML. Hemos intentado representar esas herramientas desde el Perspectiva de los profesionales

para proporcionarles información suficiente para seleccionar y utilizar una herramienta adecuada en esta búsqueda aún en curso para mejorar el rendimiento de Python. **Resultados** Hemos identificado diferentes tipos de técnicas durante nuestra encuesta que resumimos brevemente aquí:

- Bibliotecas de paralelización: MPI, OpenMP y basadas en tareas
- Bibliotecas sin cita previa
- Transformación de programas: transpiladores, JIT, compiladores generales (por ejemplo, basados en LLVM)
- Marcos completos **Alcance y limitaciones del estudio** Para mitigar el riesgo de estar sesgados en nuestra propia investigación, intentamos ser lo más abiertos posible siguiendo un proceso narrativo simple. Además, la revisión narrativa nos permite proporcionar DS y ML profesionales con una visión global de las diferentes técnicas existentes **Intérpretes de Python** CPython sirve como banco de pruebas principal para nuevas funciones y mejoras del lenguaje en Python. Esto se debe a que lo mantiene la misma comunidad que diseña el idioma. Los desafíos de rendimiento encontrados en Python están directamente asociados con la implementación de CPython.

7 CONCLUSIÓN

Nuestro artículo destacó diferentes enfoques para mejorar el rendimiento de Python con respecto a tres escenarios destinados a cubrir la mayoría de las necesidades que surgen en la práctica de DS y ML. Cada escenario cubre un estereotipo peculiar de desarrollador que se ocupa de tareas de ML y DS. Representan perfiles de profesionales que van desde una forma muy sencilla de usar Python (es decir, Python básico) hasta el uso de bibliotecas numéricas estándar, hasta el uso de grandes marcos integrados.

https://github.com/KELY04PONCE/LENGUAJE-DE-PROGRAMACION-.git