INVESTIGACIÓN JOVEN
Revista Electrónica
ISSN: 2314-3991

FACULTAD DE INFORMÁTICA

PLANIFICACIÓN Y ACELERACIÓN DE ALGORITMOS DE MACHINE LEARNING

Libutti, Leandro

De Giusti, Laura (Dir.); Naiouf, Marcelo (Codir.)

Instituto de Investigación en Informática (III-LIDI). Facultad de Informática, UNLP.

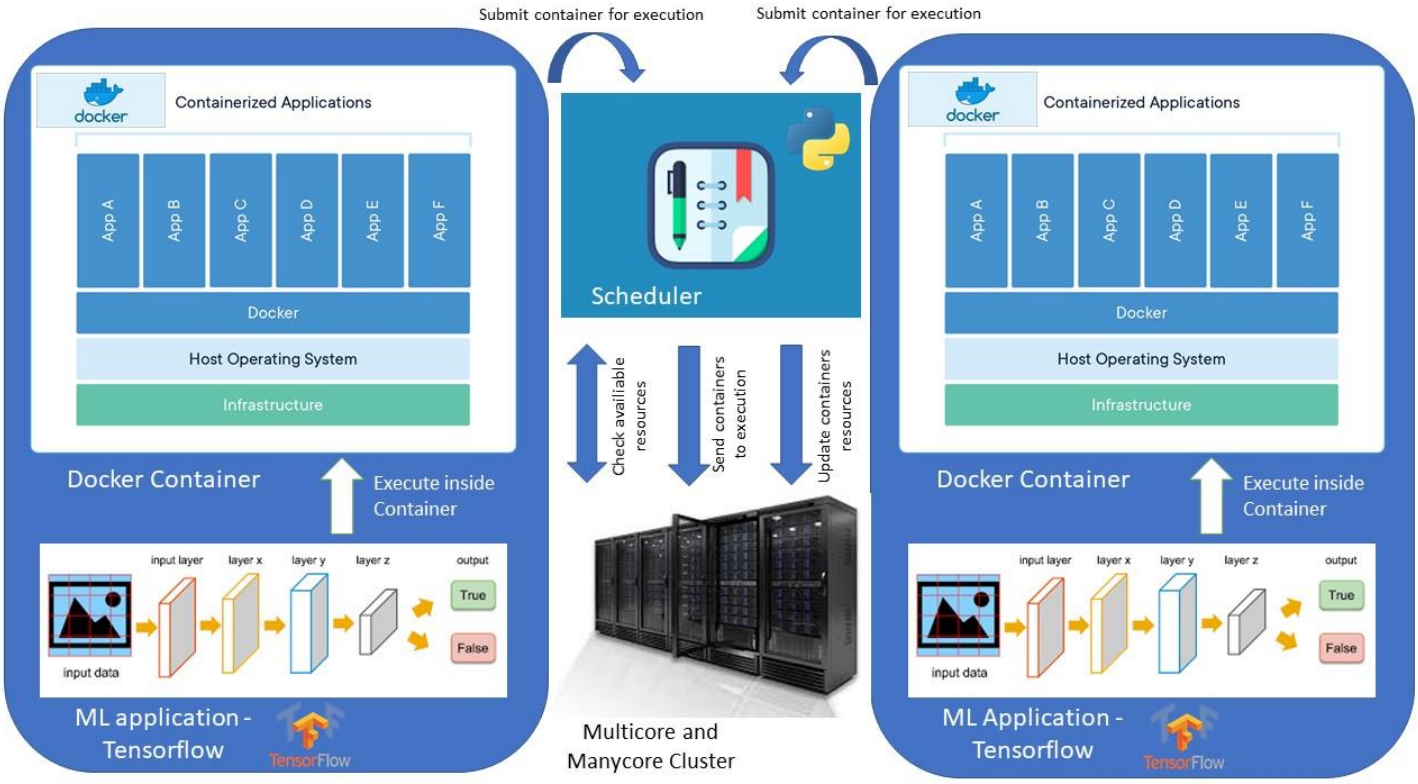
leanlibutti@gmail.com

PALABRAS CLAVE: HPC, Paralelismo Dinámico, Tensorflow, Co-Planificación, Contenedores.

OPTIMIZATION OF ENERGY CONSUMPTION IN PARALLEL ALGORITHMS

KEYWORDS: HPC, Malleability, Tensorflow, Co-Scheduling, Containers.

Resumen gráfico



INVESTIGACIÓN JOVEN
Revista Electrónica
ISSN: 2314-3991

Resumen

El crecimiento exponencial del interés del Machine Learning (ML) en la última década está relacionado con tres avances fundamentales:

1. el desarrollo de mejores algoritmos con aplicaciones directas en muchos campos de la ciencia y la ingeniería;

2. la disponibilidad de cantidades masivas de datos y la viabilidad de almacenarlos y analizarlos de manera eficiente

3. La aparición de arquitecturas de hardware novedosas, normalmente paralelas y / u homogéneas, que permiten una explotación adecuada de ambos nuevos algoritmos en grandes conjuntos de datos en un tiempo asequible.

El framework de ML denominado TensorFlow (TF) se diseñó para proporcionar capacidades de subprocesos múltiples, extendidas con soporte de acelerador de hardware para aprovechar el potencial de las arquitecturas modernas. La cantidad de paralelismo en las versiones actuales se puede seleccionar en varios niveles bajo demanda. Sin embargo, esta selección es fija y no puede variar durante la ejecución de sesiones de entrenamiento / inferencia. Esto restringe en gran medida la flexibilidad y elasticidad del framework, especialmente en escenarios en los que múltiples instancias de TF coexisten en una arquitectura paralela.

En este plan de investigación se proponen los siguientes trabajos:

• Realizar las modificaciones dentro de TF para soportar la selección dinámica de paralelismo, con el fin de brindar una maleabilidad transparente a la infraestructura.

• Integración con un co-planificacor. Una infraestructura de framework maleable solo tiene sentido real cuando se combina con un administrador de recursos de nivel superior (o co-planificador), que aprovecha la maleabilidad subyacente (en este caso dentro de TF) y de forma dinámica modifica la cantidad de recursos que se les asigna de forma coordinada.

• Creación de una API de maleabilidad. Actualmente se selecciona internamente en puntos de ejecución específicos como prueba de conceptos. Sin embargo, su gestión debe ser transparente y seleccionable externamente, bajo demanda. Una API ad-hoc para seleccionar el número de subprocesos activos / inactivos será obligatoria, junto con una infraestructura para soportar la variación de subprocesos por medio de la recepción de señales del sistema operativo.

• Gestión a través de contenedores. Los contenedores permiten una reducción dinámica del recurso en cuanto a número de núcleos, cantidad de memoria y dispositivos externos, entre otros. Sin embargo, la reducción externa del número de núcleos asignados sin una reducción adecuada de los subprocesos de software internos genera un efecto de sobre-suscripción no aceptable. Como los procesos de entrenamiento / inferencia de TF generalmente se limitan a los contenedores de Docker, es obligatorio admitir la maleabilidad en el marco. La interacción entre la gestión de recursos por contenedor y la maleabilidad en TF es, por lo tanto, un objetivo principal de nuestra investigación.

Multimedia

http://sedici.unlp.edu.ar/handle/10915/114189