Tabriz Journal of Electrical Engineering (TJEE), vol. 54, no. 2, Summer 2024 Serial no. 108
DOI: 10.22034/tjee.2023.58613.4729

Distributed Job Scheduling in on-Demand GPU as a Service Systems

Arezoo Jahani *, Leila Al-Sadat Momeni

Faculty of Electrical Engineering, Sahand University of Technology, Tabriz, Iran.
E-mails: L_momeni400@sut.ac.ir, a.jahani@sut.ac.ir
* corresponding author

Short Abstract
Optimal scheduling of resources is essential on GPU-based servers that are suitable for parallel tasks. These resources usually have a high speed and therefore have a high cost. In order to make optimal use of these resources, service providers must be able to choose the best type of virtual machine, the best type of GPU processor, and the best number of this type of processor for each request. Such a problem is called an optimization problem. The present article, while modeling the resource allocation problem as a linear optimization problem, presents a new method for distributing requests. The proposed method uses a central queue and then distributes requests among several local queues using a new request distribution method. Then it schedules and executes the tasks in each local queue in parallel. Scheduling in each local queue determines, for each request: (1) the best type of virtual machine, (2) the best type of GPU processor, and (3) the best number of GPU processors. The comparison of the proposed method with the latest available methods shows a decrease in execution time, a decrease in response time, and a significant decrease in the cost of using resources in the proposed method.

Keywords
Task scheduling, GPU-based servers, request distribution, local queuing.

1- Short Introduction

GPU servrs can run paralle jobs like machine learning traing tasks. The execution time of these types of tasks is highly variable and cannot be determined due to the possibility of changing various features. For this reason, it is usually prophesied. Scheduling tasks that do not have a definite execution time is very necessary and challenging. The method proposed in this article seeks to formulate the scheduling problem of these tasks with the aim of reducing the cost and reducing the delay time.

2- Proposed Work and Methodology

The problem of scheduling tasks related to machine learning training was investigated in this article and formulated with the aim of reducing cost and reducing delay time. In order to reduce the response time and save the used resources, a distributed scheduling method was presented. In this method, requested tasks were placed in a central queue and then gradually and using methods were transferred to local queues to be executed in a parallel and distributed manner.

3- Conclusion

GPU servrs can run paralle jobs like machine learning traing tasks. The proposed method for task scheduling was able to use less time for execution and less resources compared to the existing hierarchical method, which reduced the scheduling cost.

4- References

[1] Filippini, Federica, Marco Lattuada, Arezoo Jahani, Michele Ciavotta, Danilo Ardagna, and Edoardo Amaldi. "Hierarchical Scheduling in on-demand GPU-as-a-Service Systems." *In 2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pp. 125-132. IEEE, 2020.

[2] Jahani, Arezoo, Marco Lattuada, Michele Ciavotta, Danilo Ardagna, Edoardo Amaldi, and Li Zhang. "Optimizing on-demand gpus in the cloud for deep learning applications training." *In 2019 4th International Conference on Computing, Communications and Security (ICCCS),* pp. 1-8. IEEE, 2019.

[3] Lattuada, Marco, Eugenio Gianniti, Danilo Ardagna, and Li Zhang. "Performance prediction of deep learning applications training in GPU as a service systems." *Cluster Computing*, 1-24, 2022.

زمانبندی توزیع‌شده وظایف در سیستم‌های سرویس‌دهی مبتنی بر GPU بر حسب تقاضا

آرزو جهانی
استادیار، دانشکده مهندسی برق، دانشگاه صنعتی سهند، تبریز، ایران

لیلا السادات مومنی
دانشجوی کارشناسی ارشد، دانشکده مهندسی برق، دانشگاه صنعتی سهند، تبریز، ایران

چکیده
زمان‌بندی بهینه منابع بر روی سرورهای مبتنی بر GPU که برای وظایف موازی مناسب هستند، بسیار ضروری است. این منابع معمولاً دارای سرعت بالایی بوده و بنابراین هزینه بالایی نیز دارند. جهت استفاده بهینه از این منابع، مراکز ارائه دهنده خدمات، باید بتوانند به ازای هر درخواست، بهترین نوع ماشین مجازی، بهترین نوع پردازندهGPU و همچنین بهترین تعداد این نوع پردازنده را انتخاب نمایند. چنین مسئله‌ای، یک مسئله بهینه‌سازی نامیده می‌شود. مقاله حاضر، ضمن مدل‌سازی مسئله تخصیص منابع به عنوان یک مسئله بهینه‌سازی خطی، روش جدیدی را برای توزیع درخواست‌ها ارایه می‌دهد. روش پیشنهادی از یک صف مرکزی استفاده نموده و سپس درخواست‌ها را با استفاده از یک روش نوین توزیع درخواست، بین چندین صف محلی توزیع می‌کند. سپس وظایف موجود در هر صف محلی را به صورت موازی زمانبندی و اجرا می‌کند. زمان‌بندی در هر صف محلی، تعیین می‌کند که به ازای هر درخواست: (۱) بهترین نوع ماشین مجازی (۲) بهترین نوع پردازنده GPU و (۳) بهترین تعداد پردازنده‌های GPU کدام است. مقایسه روش پیشنهادی با آخرین روش‌های موجود، نشانگر کاهش زمان اجرا، کاهش زمان پاسخ و همچنین کاهش چشمگیر هزینه استفاده از منابع در روش پیشنهادی است.

کلمات کلیدی
زمانبندی وظایف، سرورهای مبتنی بر GPU، توزیع درخواست‌ها، صف محلی.

نام نویسنده مسئول: دکتر آرزو جهانی
ایمیل نویسنده مسئول: a.jahani@sut.ac.ir

تاریخ ارسال مقاله: ۱۴۰۲/۰۷/۰۶
تاریخ(های) اصلاح مقاله: ۱۴۰۲/۰۸/۱۶
تاریخ پذیرش مقاله: ۱۴۰۲/۰۹/۱۳

۱- مقدمه

واحد پردازش گرافیکی یا (GPU) که برای سریع‌تر انجام شدن کارهای گرافیکی طراحی شده است، می‌تواند برای پردازش موازی داده‌ها نیز استفاده گردد. بنابراین این نوع از واحدهای پردازش گرافیکی، برای سرعت بخشیدن به طیف وسیعی از بارهای کاری از جمله هوش مصنوعی و یادگیری ماشین استفاده می‌شوند و معمولاً سرعتی در حدود ۵ تا ۴۰ برابر پردازشگر معمولی دارند. اما هزینه استفاده از آنها نیز در حدود ۵ تا ۸ برابر یک CPU است [۱]. ارائه‌دهندگان خدمات رایانش ابری، با هدف ارائه خدمات بهتر و داشتن توانایی در پاسخگویی به وظایف مربوط به آموزش شبکه‌های عصبی و یادگیری ماشین، در تلاش هستند تا از سرورهای GPU بهره گیرند. این سرورها، به دلیل هزینه خریداری و نگهداری بالا، نیاز به زمانبندی بهینه جهت استفاده از منابع را دارند [۲].

بنابراین از یک سمت سرورهایی با منابع مبتنی بر پردازشگر گرافیکی با توانایی اجرای موازی وظایف را داریم و از سمت دیگر، وظایف مربوط به آموزش شبکه‌های عصبی را داریم. نقطه اشتراک این دو بخش، اجرای وظایف یادگیری ماشین بر روی سروهای مبتنی بر پردازشگر گرافیکی است. این مسئله به عنوان یک مسئله زمانبندی شناخته شده است [۳]. زمانبندی وظایف، معمولاً برای متعادل کردن بار روی سیستم، اطمینان از توزیع عادلانه منابع، و الویت‌بندی براساس قوانین تعیین شده انجام می‌شود. هدف استفاده از الگوریتم‌های زمانبندی، تضمین حداکثر استفاده از منابع، تخصیص منصفانه منابع، تضمین حداقل زمان انتظار و تضمین حداقل زمان پاسخ است [۴]. به منظور زمانبندی بهینه منابع و پاسخگویی به وظایف، مسئله مورد بررسی را به عنوان یک مسئله بهینه‌سازی خطی مدلسازی کرده و در این مقاله در پی حل آن هستیم.

ما سناریویی را فرض می‌کنیم که در آن چندین Job (که از این پس وظیفه نامیده خواهند شد) برای اجرا بر روی منابع یک ارائه‌دهنده خدمات ارسال شده است. اطلاعات اولیه در مورد وظایف، شامل زمان ورود به سیستم (submission time)، زمان سررسید (deadline) و هزینه تأخیر (tardiness weight) آنها است. ممکن است چندین وظیفه به طور همزمان وارد سیستم شوند و اطلاعات زیادی در مورد زمان تقریبی ورود وظایف وجود ندارد. در سمت ارائه‌دهنده خدمات، چندین سرور با چندین گره محاسباتی وجود دارد که هر گره محاسباتی، شامل چندین نوع ماشین مجازی مختلف به همراه تعداد زیادی پردازشگر گرافیکی به ازای هر نوع ماشین مجازی است. هر ماشین مجازی، هزینه مربوط به خود را دارد. همچنین امکان انتخاب بیش از یک نوع ماشین مجازی به ازای هر گره محاسباتی وجود ندارد. با ورود وظایف به سیستم، زمانبندی آغاز می‌گردد. هر زمانبندی، مشخص می‌نماید که ۱) بهترین گره محاسباتی کدام است. ۲) بهترین نوع ماشین مجازی کدام است. و ۳) بهترین تعداد پردازشگر گرافیکی برای هر وظیفه کدام است. زمانبندی وظایف، در ابتدای شروع به کار سیستم

آغاز شده و زمانبندی مجدد در یکی از حالات زیر اتفاق می‌افتد. ۱) حداقل یک وظیفه جدید وارد سیستم گردد. ۲) حداقل یکی از وظایف قبلی به اتمام برسد. ۳) تا یک مدت زمان مشخصی، هیچ یک از دو مورد قبلی اتفاق نیفتند و این زمان مشخص سپری شده باشد. در هر زمانبندی مجدد، تمام وظایف قبلی نیز تا جایی که اجرا شده‌اند، متوقف شده و سپس مجدداً زمانبندی می‌گردند.

مشکل اساسی در روش‌های پیشین زمانبندی عبارتند از، عدم توجه به زمان مورد نیاز برای تکمیل فرآیند زمانبندی که خود این امر موجب افزایش زمان پاسخ سیستم و نزدیک شده به زمان سررسید وظایف می‌گردد. مشکل دیگر مربوط به نحوه مدلسازی سیستم و ساخت مدل بهینه‌سازی است که بتوان وظایف را به درستی و در سریعترین زمان ممکن زمانبندی کرده و نتایج را مورد استفاده قرار داد [۵]. در این مقاله، قصد داریم مورد اول را با تبدیل زمانبندی به یک زمانبندی سلسله مراتبی آدرس‌دهی کنیم. زمانبندی سلسله مراتبی، به مفهوم تقسیم سرورهای ارائه‌دهنده خدمات به چندین گروه است که هر گروه، دارای یک صف محلی از درخواست‌ها است و که توسط یک صف مرکزی، تعیین تکلیف می‌گردند. همچنین قصد داریم، مشکل دوم را با مدلسازی مسئله به صورت یک مسئله بهینه‌سازی خطی، به ازای هر صف محلی آدرس‌دهی کنیم. نحوه توزیع درخواست‌ها از صف مرکزی به صف‌های محلی و همچنین نحوه زمانبندی بهینه در هر صف محلی در ادامه بیان شده است.

سازماندهی مقاله به این صورت است. بخش دوم، روش‌های پیشین در رابطه با زمانبندی وظایف بر سرورهای مبتنی بر GPU را به تفصیل بیان می‌کند. بخش سوم، روش پیشنهادی برای توزیع درخواست‌ها و مدلسازی زمانبندی را توضیح می‌دهد. بخش چهارم، شامل معرفی داده‌های مورد استفاده و نحوه شبیه‌سازی مسئله و ارزیابی کارایی آن است. در نهایت بخش پنجم، نتیجه‌گیری مقاله را انجام داده و تعدادی از کارهای آتی را معرفی می‌کند.

۲- کارهای پیشین

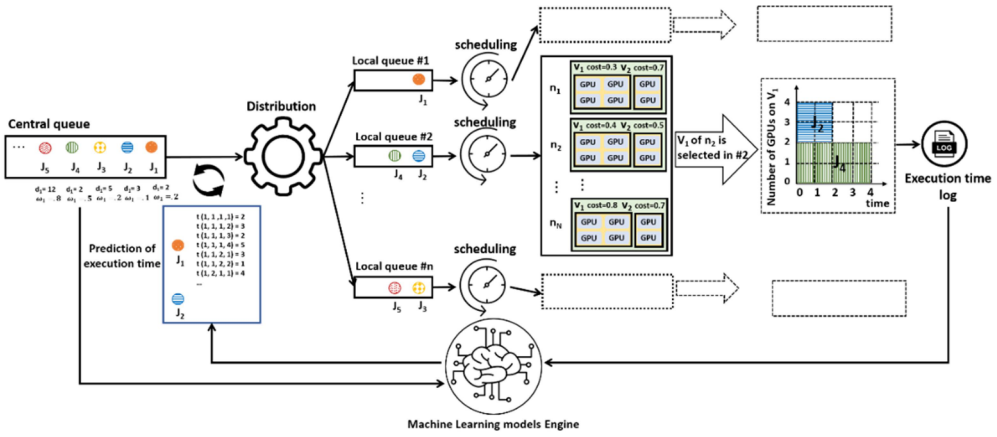
با توجه به سرعت بالای پردازش در پردازنده‌های گرافیکی، می‌توان از آنها برای اجرای موازی چندین کار به صورت همزمان استفاده کرده و زمان اجرا را به طور قابل توجهی کاهش داد. همچنین امکان استفاده از چندین پردازنده گرافیکی برای یک وظیف همنفرد نیز وجود دارد. اما پیچیدگی کسئله زمان یافزایش پیدا می‌کند که قصد داشته باشیم زمانبندی این منابع را در سیستم‌های ارائه دهنده خدمات ابری و برای درخواست‌هایی که به طور پیوشته به سیستم وارد می‌شوند، انجام دهیم. در سیستم‌های رایانش ابری، به دلیل اینکه امکان داشتن چندین نوع ماشین مجازی مختلف با تعداد متفاوتی از پردازنده گرافیکی وجود دارد، زمانبندی دشوار می‌گردد. در این بخش قصد داریم روش‌های موجود برای زمانبندی سیستم‌های سرویس‌دهنده GPU براساس تقاضا را مورد بررسی قرار داده و تحلیل نماییم.

پژوهش‌های زیادی در زمینه زمانبندی GPU برای سیستم‌هایی با محاسبات با کارایی بالا (HPC) انجام شده است که عمده هدف آنها تعادل بار و استفاده بهینه از هر دوی CPU و GPU بوده است [۶]. اما پژوهش‌های اندکی مستقیماً به سرورهای مختص وظایف هوش مصنوعی پرداخته‌اند. در اغلب مقالات زمانبندی، معمولاً تمرکز بر روی تخصیص منابع فقط در شروع اجرای وظایف است. الگوریتم‌هایی ارایه می‌شوند و این الگوریتم‌ها با رسیدن وظایف جدید، منابع خالی سیستم را شناسایی کرده و شروع به زمانبندی با اهداف تعیین شده می‌کنند. منابع تخصیص داده شده، فقط پس از اتمام وظایف، آزاد شده و امکان استفاده مجدد از آنها فراهم می‌گردد. همچنین اهداف زمانبندی می‌تواند، کاهش هزینه، افزایش نرخ پذیرش، کاهش زمان پاسخگویی، تعادل بار موجود بر روی سرورها و یا ترکیبی از این موارد باشد [۷-۸].

معمولاً ابتدا سیاست‌هایی برای زمانبندی تعریف شده و سپس الگوریتم اجرا می‌گردد. سیاست استفاده از GPU با توجه به برخی وابستگی‌های پسین و پیشین وظایف در مقاله [۹] بررسی شده است. در پژوهشی دیگر، تعداد هسته‌های مورد نیاز برای هر وظیفه، بر اساس اولویت آن وظیفه تعیین می‌گردد [۱۰]. یک روش مبتنی بر برنامه‌ریزی محدودیت‌ها در [۱۱] ارائه شده است که در تلاش است وظایف بزرگ را با توجه به بودجه کاربر و هزینه وظایف زمانبندی کند. روش دیگری در [۱۲] ارائه شده است که تمرکز اصلی آن برآورد کردن زمان سررسید وظایف است.

در پژوهش [۱۳]، بنچمارکی ارائه شده است که توانایی استفاده از زمانبندی‌های موجود را در تمام انواع سیستم‌عامل‌های مختلف فراهم می‌آورد. در واقع این روش، دارای قابلیت حمل بالایی بوده و به راحتی بر روی پلتفرهای مختلف می‌تواند انواع تکنیک‌های زمانبندی موجود را ارائه دهد. اما با وجود تمام مزایای ذکر شده، مصرف انرژی که مسئله مهمی در مراکز داده است را بررسی نکرده است. یک روش زمانبندی به نام Mystic در [۱۴] ارائه شده است که روشی آگاه از تداخل منابع و وظایف است و برای اجرای مشترک وظایف در خوشه‌های مبتنی بر GPU و سرورهای ابری استفاده می‌گردد.

هدف اصلی مقاله حاضر، زمانبندی وظایف در سیستم‌هایی است که ارائه دهنده خدمات GPU برحسب تقاضا هستند. سرورهای موجود در این نوع ارائه‌دهندگان خدمات، دارای چندین نوع مختلف از ماشین مجازی بوده و هر ماشین مجازی، امکان استفاده از تعداد مشخصی پردازنده گرافیکی را دارد. در این زمینه، پژوهشی ارائه شده است که تمام وظایف موجود را در یک صف مرکزی قرار داده و سپس با استفاده از یک مدل بهینه‌سازی خطی، وظایف را جهت اجرا بر روی بهترین سرور از مجموعه سرورهای موجود، بهترین ماشین مجازی و بهترین تعداد پردازنده‌های گرافیکی زمانبندی می‌کند [۱۵]. در روش ارائه شده توسط این نویسندگان، به دلیل وجود فقط یک صف، امکان سرویس‌دهی به تعداد محدودی از وظایفی که در یک بازه زمانی وارد سیستم می‌شوند، وجود دارد. اگر تعداد درخواست‌ها بیشتر باشد، خود الگوریتم زمانبندی دچار مشکل شده و پاسخگویی آن به مدت زمان بسیار زیادی نیاز خواهد داشت. مشکل مطرح شده در این پژوهش [۱۵]، توسط پژوهشی دیگری تحلیل گردیده است. در مقاله [۵] به منظور افزایش توانایی الگوریتم زمانبندی برای پاسخگویی به تعداد بیشتری از درخواست‌های همزمان، یک روش سلسله مراتبی ارائه شده است که در آن، سرورهای ارائه دهنده خدمات، به چندین گروه کاملاً یکسان تقسیم می‌گردند که هر گروه از این سرورها، دارای یک صف محلی است. در ابتدا تمام وظایف در یک صف مرکزی قرار دارند و پیش از اجرا، توسط یک الگوریتم توزیع درخواست، به صف‌های محلی اختصاص داده می‌شوند. سپس وظایف موجود در هر صف محلی، زمانبندی شده و بر روی منابع همان صف اجرا می‌گردد. این الگوریتم توزیع شده، مشکل قبلی را حل می‌کند. در این پژوهش، یک مدل برنامه‌ریزی خطی عدد صحیح (MILP) برای مدیریت وظایف در یک خوشه از سرورهای مبتنی بر GPU پیشنهاد شده است. روش پیشنهادی می‌تواند تاریخ سررسید وظایف را رعایت نماید و در صورت گذر از تاریخ سررسید، یک هزینه جریمه به آنها اضافه می‌گردد. ایراد چنین روشی، وجود یک الگوریتم زمانبندی مرکزی برای کنترل تمام صف‌های محلی به صورت کاملاً متمرکز است. همچنین در بخش توزیع درخواست‌ها، فقط روش نوبت گردشی استفاده شده است که باعث می‌گردد، گاهی اوقات وظایفی با تاریخ سررسید یکسان یا نزدیک، در یک صف محلی قرار گرفته و در زمان اجرا به ناچار، جریمه دریافت کنند. مقاله حاضر قصد دارد مشکلات ارائه شده در دو پژوهش اخیر را با ارائه روشی نوین در توزیع درخواست‌ها و روشی برای مدلسازی و حل مسئله، هموار نماید.



شکل ۱- شمای کلی روش پیشنهادی

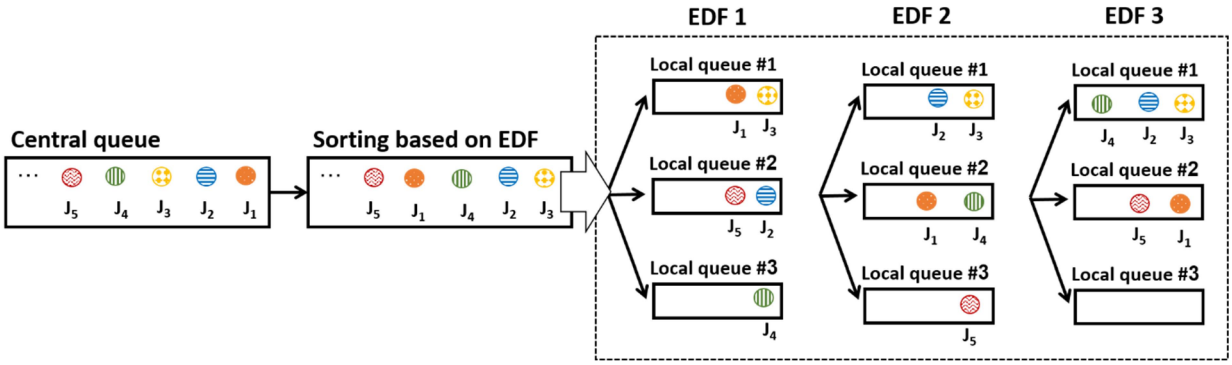
۳- روش پیشنهادی

این بخش از مقاله، مفروضات و سیستم مدل استفاده شده را شرح داده و سپس روش توزیع درخواست و مدلسازی مسئله را با جزئیات کامل بیان می‌کند. سیستم مدل استفاده شده در مقاله فرض می‌کند که تعدادی سرور ارائه‌دهنده خدمات با شرایط مشخص وجود دارد. هر سرور دارای چندین نوع مختلف ماشین مجازی است و هر ماشین مجازی دارای چندین پردازنده گرافیکی است. هر نوع ماشین مجازی، یک هزینه استفاده و یک ظرفیت دارد. فرض بر این است که امکان دسترسی به همه انواع مختلف ماشین‌های مجازی توسط تمام سرورها وجود دارد. اما در هر بار استفاده از سرور، امکان انتخاب فقط یکی از انواع مختلف ماشین مجازی وجود دارد. درخواست‌ها از طریق یک صف مرکزی به این سیستم ارسال می‌شوند. این فرآیند در شکل ۱ نشان داده شده است. ممکن است در هر بازه زمانی، چندین درخواست به صورت همزمان وارد سیستم شوند. هر درخواست، شامل یک تاریخ سررسید و یک جریمه دیرکرد است. در صورتی که زمان اتمام درخواست پس از زمان سررسید باشد، جریمه دیرکرد، به هزینه نهایی اضافه خواهد شد. بنابراین زمان ورود درخواست‌ها که در این مقاله وظیفه و در بخش مدلسازی job نامیده شده است، مشخص است. اما اطلاعات دیگری نظیر میزان زمان و میزان منابع مورد نیاز به ازای هر درخواست موجود نیست. عدم وجود اطلاعات در مورد منابع و زمان مورد نیاز، چالش اساسی این مسئله است. زیرا در صورتی که زمان مورد نیاز برای وظایف از ابتدا مشخص بود، امکان زمانبندی وظایف با روش‌های بهینه‌سازی قطعی امکانپذیر بوده و سریعتر نتیجه حاصل می‌شود. اما چون وظایف از نوع وظایف آموزشی یادگیری ماشین هستند که در آنها، انواع مختلفی از شبکه‌های عصبی با انواع و تعداد مختلفی از لایه‌های پنهان استفاده شده است، تعیین مدت زمان مورد نیاز برای اتمام اجرا به سادگی قابل محاسبه نیست.

همان‌طور که در شکل ۱ نشان داده شده است، در این مقاله برای حل مسئله مربوط به عدم اطلاع از زمان مورد نیاز برای اتمام وظایف، از یک موتور پیشگویی زمان اجرا استفاده شده است. در واقع، فرض بر این است که اطلاعات زمان اجرای وظایف پیشین ثبت شده است. سپس یک موتور حاوی الگوریتم‌های یادگیری ماشین برای پیشگویی زمان اجرای وظایف جدید وجود دارد. این موتور پیشگویی از روش پیشنهادی در مرجع [۱۶] استخراج شده است و می‌تواند زمان اجرای وظایف جدید را براساس انواع مختلف ماشین‌های مجازی و تعداد متفاوتی از پردازنده‌های گرافیکی پیشگویی کند. به عنوان نمونه می‌تواند مشخص نماید که درخواست شماره ۱ (J_1) در صورتی که بر روی سرور شماره یک، با نوع ماشین مجازی دو و تعداد یک پردازنده گرافیکی ((t(1,1,1,1)) اجرا گردد، چه مدت زمانی نیاز خواهد داشت. در صورتی که همان درخواست، بر روی سرور شماره ده، با نوع ماشین مجازی سه و تعداد دو پردازنده گرافیکی ((t(1,10,3,2)) اجرا گردد، چه مدت زمانی نیاز خواهد داشت و به همین ترتیب پیشگویی را برای انواع مختلف حالات ارائه می‌دهد. در واقع پیشگویی‌های انجام شده، دقیق نیستند و به همین دلیل در زمان اجرای وظایف، در مواقع مورد نیاز که زمانبندی مجدد صورت می‌گیرد، زمان باقیمانده واقعی اجرای وظایف مجدداً محاسبه می‌گردد.

وظیفه الگوریتم زمانبندی به این صورت است که مشخص نماید به ازای هر درخواست، ۱) بهترین نوع ماشین مجازی، ۲) بهترین نوع پردازنده و ۳) بهترین تعداد پردازنده گرافیکی کدام است. در مبحث زمانبندی، دو مسئله تحت عناوین زمانبندی (scheduling) و زمانبندی مجدد (rescheduling) وجود دارد. زمانبندی، برای تعیین زمان شروع اجرا و میزان منابع تخصیص یافته استفاده می‌شود. اما زمانبندی مجدد، برای تعیین تغییر زمانبندی قبلی، پیش از اتمام اجرا، بنا بر شرایط موجود است. بنابراین در این مقاله قصد داریم، بخش زمانبندی مجدد را نیز تحت یکی از شرایط زیر اجرا کنیم: ۱) رسیدن حداقل یک درخواست جدید. ۲) اتمام اجرای حداقل یکی از درخواست‌های موجود و ۳) گذر یک مدت زمان تعیین شده بدون رخداد هیچ یک از دو حالت قبلی.

روش پیشنهادی در شکل ۱ دارای دو مرحله مجزا می‌باشد. (۱) بخش توزیع درخواست‌ها و (۲) بخش زمانبندی درخواست‌ها. ابتدا روشی برای توزیع درخواست‌ها از صف مرکزی به صف‌های محلی پیشنهاد شده است تا امکان اجرای موازی چندین درخواست به صورت همزمان فراهم گردد. در حالت غیرسلسله‌مراتبی، تمام وظایف به صورت متمرکز به صف محلی ارسال می‌گردد که موجب افزایش زمان اجرا و زمان پاسخ می‌گردد. اما در روش غیرسلسله‌مراتبی، هر صف محلی مسئول منابع محلی خود است. با این روش، در حضور تعداد زیادی درخواست، امکان پاسخگویی به درخواست‌ها در زمان کمتری فراهم خواهد گردید. سپس به ازای صف‌های محلی، یک روش زمانبندی ارایه شده است. مسئله زمانبندی به صورت یک مدل برنامه‌ریزی خطی عدد صحیح (MILP) مدلسازی شده و توسط روش glpk حل شده است. این دو مرحله در ادامه با جزئیات شرح داده شده است. در نمونه ترسیم شده در شکل ۱، فرض بر این است که پنج درخواست در یک بازه زمانی وارد سیستم شده است. در این ارائه دهنده، سه صف محلی وجود دارد. ابتدا درخواست‌ها از صف مرکزی، با استفاده از یک روش توزیع درخواست، به سه صف محلی انتقال یافته‌اند. سپس در هر صف محلی زمانبندی شده‌اند. در صف محلی دو، دو درخواست شماره ۲ و ۴، برای اجرا بر روی گره شماره ۲ انتقال پیدا کرده‌اند. این وظایف بر روی ماشین مجازی نوع یک که دارای ۴ پردازنده گرافیکی است، اجرا می‌شوند. J_2 دو پردازنده دریافت کرده است و پس از دو واحد زمانی به اتمام رسیده است. J_4 نیز دو پردازنده گرافیکی دریافت کرده است، اما پس از چهار واحد زمانی به اتمام رسیده است.



شکل ۲- سه سناریوی مختلف توزیع درخواست

۳-۱- *توزیع درخواست‌ها (Distributaion)*

برای توزیع درخواست‌های موجود در صف مرکزی به صف‌های محلی، می‌توان از روش‌های متفاوتی استفاده کرد. در مقاله [۵] از روش نوبت گردشی (RR)[۵] برای توزیع درخواست‌ها استفاده شده است. با توجه به اینکه ارائه روش سلسله مراتبی برای زمانبندی درخواست‌ها که در این مقاله نیز از چنین روشی استفاده شده است، اولین بار در مقاله [۵] ارایه شده است، بنابراین روش‌های دیگر توزیع درخواست بررسی نشده است. مشکل اصلی توزیع نوبت گردشی، عدم توجه به تاریخ سررسید وظایف است و ممکن است چندین درخواست با تاریخ سررسید یکسان و یا نزدیک به همدیگر به یک صف محلی هدایت شوند. بنابراین به منظور حل این مشکل، در این مقاله، پیش از توزیع درخواست‌ها به صورت نوبت گردشی، ابتدا درخواست‌هایی که در یک بازه زمانی خاص وارد سیستم می‌شوند، براساس ترتیب صعودی تاریخ سررسید مرتب شده و سپس با استفاده از سه سناریو مختلف بین صف‌های محلی توزیع می‌شوند. سناریو اول نوبت گردشی RR است. در سناریو دوم و سوم، وظایف همچنان به صورت نوبت گردشی ولی در هر بار، به ترتیب دو و سه درخواست به یکی از صف‌های محلی منتقل می‌گردد. شکل ۲، سه سناریوی پیشنهادی را با یک مثال نشان می‌دهد.

همان‌طور که در شکل ۲ نشان داده شده است، ابتدا درخواست‌های موجود در صف مرکزی، براساس ترتیب صعودی تاریخ سررسید (EDF)[۵۵] مرتب می‌شوند. سپس در سناریوی یک (EDF1)، درخواست‌ها به صورت نوبت گردشی توزیع می‌شوند. در سناریوی دو (EDF2)، درخواست‌ها به صورت نوبت گردشی، ولی در هر بار گردش، دو درخواست توزیع می‌شود. نهایتاً در سناریوی سوم (EDF3)، درخواست‌ها به صورت نوبت گردشی، ولی در هر بار گردش، سه درخواست توزیع می‌شود. البته نحوه مرتب‌سازی درخواست‌ها نیز بسته به نظر طراح می‌تواند یکی از این سه حالت باشد. این سناریوها به منظور توزیع بهینه درخواست‌ها در صف‌های محلی است، هدف سناریوی اول، این است چندین درخواست با تاریخ سررسید یکسان وارد یک صف محلی نشوند. هدف سناریوی دوم و سوم این است که تعداد محدودی درخواست با تاریخ سررسید یکسان وارد سیستم شوند تا بتوانند از ماشین مجازی یکسانی استفاده نمایند تا میزان هزینه نهایی استفاده از منابع برای تامین کننده منابع، کاهش یابد. با وجود اینکه توزیع صددرصدی درخواست‌ها، منجر به استفاده بهینه از منابع گشته و زمان اجرا و بالطبع آن زمان پاسخگویی را کاهش می‌دهد، اما هزینه استفاده از منابع را افزایش می‌دهد. بنابراین هدف، بررسی روش مرتب‌سازی درخواست‌ها براساس تاریخ سررسید، پیش از توزیع درخواست و زمانبندی است. تعداد صف‌های مورد نیاز و بهینه نیز در بخش ارزیابی، مورد بررسی قرار خواهد گرفت.

۳-۲- *زمانبندی (Scheduling)*

برای زمانبندی وظایف در هر صف محلی، یک الگوریتم بهینه‌سازی و مدلسازی مسئله مورد نیاز است. الگوریتم زمانبندی، به ازای هر صف محلی جداگانه اجرا می‌گردد، اما کاملاً دارای شرایط یکسانی است. اما می‌تواند ورودی و منابع متفاوتی داشته باشد. هر چند در این مقاله فرض بر این است که تمام صف‌های محلی، دارای منابعی با تعداد و شرایط یکسان هستند. اما امکان متفاوت بودن تعداد و شرایط منابع وجود دارد و صرفاً لازم است الگوریتم بهینه‌سازی به ازای شرایط هر صف محلی، تنظیم گردد.

مدلسازی مسئله به صورت یک مسئله برنامه‌ریزی خطی عدد صحیح در ادامه توضیح داده شده است. هدف و یا تابع اصلی مسئله به این صورت است: ۱) کمینه‌سازی اختلاف زمان اجرا و تاریخ سررسید وظایف (بطوریکه حدالامکان هیچ وظیفه‌ای، دیرتر از تاریخ سررسید خود به اتمام نرسد)، ۲) کمینه‌سازی پردازنده‌های گرافیکی بلااستفاده، ۳) کمینه‌سازی هزینه ماشین‌های مجازی انتخاب شده، و ۴) کمینه‌سازی هزینه‌ی مربوط به درخواستی که زودتر از بقیه به اتمام می‌رسد (در واقع سریعترین درخواست است).

زمانی که اهداف یک مسئله برنامه‌ریزی، بیش از یک مورد باشد، می‌توان با استفاده از روش‌هایی، این اهداف را باهمدیگر ترکیب کرد. البته ترکیب اهداف، به کمینه و یا بیشینه بودن توابع آنها بستگی دارد. در مسئله حاضر تمام توابع هدف از نوع کمینه‌سازی می‌باشند، بنابراین با روش وزندهی یکسان، تمام اهداف به یک هدف واحد تبدیل شده‌اند. تمام پارامترها و متغیرهای استفاده شده در مدلسازی مسئله در جدول ۱ نشان داده شده است.

مطابق با جدول ۱، یک مجموعه وظایف J از طریق صف مرکزی وارد سیستمی می‌گردد که دارای $K = \{1, \dots, k\}$ صف محلی است. مجموعه درخواست‌های مربوط به هر صف محلی برابر با J_k است. فرض بر این است که ویژگی‌های منابع موجود به ازای هر صف محلی کاملاً یکسان هستند و هر صف محلی به N_k گره یا سرور دسترسی دارد که هر گره دارای V نوع مختلف از ماشین‌های مجازی است و هر ماشین مجازی بسته به نوع خود، دارای چندین پردازنده GPU است که می‌تواند به ازای هر درخواست، تعدادی از این پردازنده‌ها را استفاده نماید. امکان اجرای چندین درخواست بر روی یک نوع ماشین مجازی با تسهیم پردازنده‌ها نیز امکان‌پذیر است. G_v مجموعه‌ای از تعداد پردازنده‌های موجود بر روی ماشین مجازی v است. به عنوان نمونه اگر یک ماشین مجازی دارای ۴ عدد پردازنده GPU باشد، مجموعه پردازنده‌های آن برابر است با $G_v = \{1,2,3,4\}$ و اگر بیش از یک درخواست برای اجرا بر روی این ماشین مجازی انتخاب شده باشد، باید مجموع پردازنده‌های تخصیص یافته با آن درخواست‌ها برابر با تعداد کل پردازنده‌های موجود باشد. مدلسازی پیشنهادی برای مسئله در مدل P1 نشان داده شده است.

جدول ۱- لیست پارامترها و متغیرهای مسئله (وظیفه و درخواست در کل مقاله به یک مفهوم هستند.)

پارامترهای مسئله	
J	مجموعه وظایف موجود در صف مرکزی
J_k	مجموعه وظایف موجود در صف محلی k
K	تعداد صف‌های محلی که هر صف با اندیس k نمایش داده می‌شود.
N	مجموعه گره‌ها یا سرورها
N_k	مجموعه گره‌ها یا سرورهای صف محلی k
V	مجموعه انواع مختلف ماشین‌های مجازی
G_v	مجموعه GPU موجود بر روی ماشین مجازی v
s_v	تعداد GPU به ازای هر ماشین مجازی v
c_v	هزینه ماشین مجازی v به ازای هر واحد زمانی استفاده
d_j	تاریخ سررسید درخواست j
ω_j	وزن مربوط به جریمه دیرکرد درخواست j
M_j	بیشترین زمان اجرای پیشگویی شده برای درخواست j
t_{jnvg}	زمان اجرای درخواست j، بر روی سرور n، ماشین مجازی v و به تعداد g عدد GPU
$\widehat{M_j}$	بیشترین هزینه‌ی مربوط به درخواست j
H	بازه زمانی بعدی برای زمانبندی
μ	جریمه برای GPU های موجود و بلااستفاده
ρ	جریمه برای درخواستی که از تاریخ سررسید آن گذشته است.
متغیرهای مسئله	
w_n^k	متغیر باینری (۱، اگر گره یا سرور n از صف k انتخاب شده باشد.) (۰، اگر گره یا سرور n از صف k انتخاب نشده باشد.)
y_{nv}^k	متغیر باینری (۱، اگر ماشن مجازی نوع v از گره یا سرور n از صف k انتخاب شده باشد.) (۰، اگر ماشن مجازی نوع v از گره یا سرور n از صف k انتخاب شده باشد.)
z_j^k	متغیر باینری (۱، اگر درخواست j از صف k برای اجرا انتخاب شده باشد.) (۰، اگر درخواست j از صف k برای اجرا انتخاب نشده باشد.)
x_{jnvg}^k	متغیر باینری (۱، اگر درخواست j از صف k برای اجرا بر روی سرور n، با ماشین مجازی نوع v و g تا GPU انتخاب شده باشد.) (۰، اگر درخواست j از صف k برای اجرا بر روی سرور n، با ماشین مجازی نوع v و g تا GPU انتخاب نشده باشد.)
θ_j^k	میزان تاخیر درخواست j از صف k
$\widehat{\theta_j^k}$	بیشترین میزان تاخیر درخواست j از صف k براساس زمان‌های پیشگویی شده
π_{jn}^k	هزینه سرور n از صف k برای اجرای درخواست j
α_{jn}^k	متغیر باینری (۱، اگر درخواست j از صف k و سرور n اولین درخواستی است که به اتمام می‌رسد.) (۰، اگر درخواست j از صف k و سرور n اولین درخواستی نیست که به اتمام می‌رسد.)

همان‌طور که در مدل p1 نشان داده شده است، تابع هدف دارای چهار بخش است: ۱) کمینه‌سازی اختلاف زمان اجرا و تاریخ سررسید وظایف، ۲) کمینه‌سازی پردازنده‌های گرافیکی بلااستفاده (μ یک ثابت مثبت و نشانگر ضریب لاگرانژ است)، ۳) کمینه‌سازی هزینه ماشین‌های مجازی انتخاب شده، و ۴) کمینه‌سازی هزینه‌ی مربوط به درخواستی که زودتر از بقیه به اتمام می‌رسد، زیرا با اتمام آن درخواست، زمانبندی مجدد باید انجام گیرد.

با توجه به کمینه‌سازی تابع هدف، تنها اولین درخواستی که در هر گره تکمیل می‌شود، دارای $\alpha_{jn}^k = 1$ خواهد بود و کارهای دیگر با $\alpha_{jn}^k = 0$ مشخص می‌شوند زیرا سریع‌ترین کار دارای کوچکترین π_{jn}^k است. در نهایت، توجه داشته باشید که ما از هزینه‌های پیکربندی مجدد گره‌های در حال اجرا غفلت می‌کنیم، زیرا این امر به چند دقیقه نیاز دارد در حالی که کارهای آموزشی یادگیری ماشین برای چندین ساعت (یا چند روز) اجرا می‌شوند. بنابراین بخش چهارم تابع هدف $\alpha_{jn}^k \pi_{jn}^k$ موجب غیرخطی شدن مسئله شده است. اما با توجه به اینکه قصد داریم از solver هایی نظیر CPLEX برای حل مسئله استفاده نماییم و معمولاً solverهای غیرخطی که از متغیرهای عدد صحیح پشتیبانی می‌کنند کمتر از حل‌کننده‌های MILP موثر هستند. بنابراین مدل P1 در بخش بعدی، خطی‌سازی خواهد شد.

محدودیت (p1b) نشان می‌دهد که به ازای هر گره از صف محلی k ($n \in N_k$)، اگر آن گره برای اجرا انتخاب شده است، فقط یک نوع ماشین مجازی مجاز است انتخاب گردد. محدودیت (p1c) نشان می‌دهد که اگر وظیفه‌ای برای اجرا انتخاب شده است، فقط بر روی یک نوع ماشین مجازی اجرا گردد و آن ماشین مجازی هم در حالت انتخاب شده باشد. محدودیت (p1d) نشان می‌دهد که اگر وظیفه‌ای برای اجرا انتخاب شده است، اطمینان حاصل گردد که خود آن وظیفه نیز در حالت انتخاب برای اجرا باشد. محدودیت‌های (p1e) و (p1f) به ترتیب نشان می‌دهند که به ازای هر وظیفه، حداقل یک پردازنده انتخاب شده باشد و در بهترین حالت، تمام وظایف موجود در صف محلی برای اجرا انتخاب شده باشند. محدودیت (p1g) تخصیص وظایف را فقط به گره‌های انتخاب شده مجاز می‌داند.

$$min \sum_{j\epsilon J_k} \omega_j \left(\theta_j^k + \rho\widehat{\theta_j^k}\right) + \mu \sum_{\substack{n\epsilon N_k \\ v\epsilon V}} (s_v y_{nv}^k - \sum_{\substack{j\epsilon J_k \\ g\epsilon G_v}} g x_{jnvg}^k) + \sum_{\substack{j\epsilon J_k \\ n\epsilon N_k \\ v\epsilon V \\ g\epsilon G_v}} \frac{g}{s_v} c_v\, t_{jvg} x_{jnvg} + \sum_{\substack{j\epsilon J_k \\ n\epsilon N_k}} \alpha_{jn}^k \pi_{jn}^k \quad \text{(p1a)}$$

Subject to:

$$\sum_{v\in V} y_{nv}^k = w_n^k; \qquad \forall\, n \in N_k \quad \text{(p1b)}$$

$$x_{jnvg}^k \le y_{nv}^k; \qquad \forall j \in J_k,\ \forall n \in N_k,\ \forall\, v \in V,\ \forall\, g \in G_v \quad \text{(p1c)}$$

$$x_{jnvg}^k \le z_j^k; \qquad \forall j \in J_k,\ \forall n \in N_k,\ \forall\, v \in V,\ \forall\, g \in G_v \quad \text{(p1d)}$$

$$\sum_{j\in J_k} z_j^k \le s_v; \quad \text{(p1e)}$$

$$\sum_{j\in J_k} z_j^k \le |J_k|; \quad \text{(p1f)}$$

$$\sum_{v\in V}\sum_{g\in G_v} x_{jnvg}^k \le w_n^k; \qquad \forall j \in J_k,\ \forall n \in N_k \quad \text{(p1g)}$$

$$\sum_{n\in N_k}\sum_{v\in V}\sum_{g\in G_v} x_{jnvg}^k = z_j^k; \qquad \forall j \in J_k \quad \text{(p1h)}$$

$$\sum_{j\in J_k}\sum_{g\in G_v} g x_{jnvg}^k \le s_v; \qquad \forall n \in N_k,\ \forall\, v \in V \quad \text{(p1i)}$$

$$\sum_{n\in N_k}\sum_{v\in V}\sum_{g\in G_v} t_{jnvg}\, x_{jnvg}^k \le d_j + \theta_j^k; \qquad \forall j \in J_k \quad \text{(p1j)}$$

$$\left(H + M_j\right)\left(1 - Z_j^k\right) \le d_j + \widehat{\theta_j^k}; \qquad \forall j \in J_k \quad \text{(p1k)}$$

$$\sum_{v\in V}\sum_{g\in G_v} t_{jnvg}\, c_v\, x_{jnvg}^k \le \pi_{jn}^k; \qquad \forall j \in J_k,\ \forall n \in N_k \quad \text{(p1l)}$$

$$\sum_{j\in J_k} \alpha_{jn}^k = w_n^k; \qquad \forall n \in N_k \quad \text{(p1m)}$$

$$\alpha_{jn}^k \le z_j^k; \qquad \forall j \in J_k,\ \forall n \in N_k \quad \text{(p1n)}$$

$$\sum_{n\in N_k} w_n^k = min\,\{|N_k|, |J_k|\} \quad \text{(p1o)}$$

$$y_{nv}^k \in \{0,1\}, z_j^k \in \{0,1\}, \alpha_{jn}^k \in \{0,1\}; \qquad \forall n \in N_k,\ \forall\, v \in V \quad \text{(p1p)}$$

$$x_{jnvg}^k \in \{0,1\}; \qquad \forall j \in J_k,\ \forall n \in N_k,\ \forall\, v \in V,\ \forall\, g \in G_v \quad \text{(p1q)}$$

$$\theta_j^k \ge 0, \widehat{\theta_j^k} \ge 0; \qquad \forall\, j \in J_k \quad \text{(p1r)}$$

$$\pi_{jn}^k \ge 0; \qquad \forall j \in J_k,\ \forall n \in N_k \quad \text{(p1s)}$$

محدودیت (p1h) نشان می‌دهد، اگر وظیفه‌ای برای اجرا انتخاب شده است، حتماً به آن یک گره، یک نوع ماشین مجازی و تعدادی پردازنده گرافیکی تخصیص یافته باشد. محدودیت (p1i) نشان می‌دهد که اگر یک نوع ماشین مجازی از یک گره انتخاب شده است ($\forall n \in N_k,\ \forall\, v \in V$)، آنگاه مجموع تمام پردازنده‌های استفاده شده از آن نوع ماشین مجازی، حداکثر به اندازه تعداد کل پردازنده‌های موجود در آن نوع باشد. محدودیت (p1j) نشان می‌دهد که به ازای هر وظیفه ($\forall j \in J_k$)، زمان اجرای مورد نیاز برای آن در زمانبندی جاری، حداکثر به اندازه زمان سررسید آن به اضافه تاخیر مورد نیاز برای اجرای آن در صف k باشد. هدف از این محدودیت، یافتن مقدار کمینه برای تاخیر اجرای وظایف است. محدودیت (p1k) نشان می‌دهد که اگر یک وظیفه در این بازه زمانی در حال اجرا نمی‌باشد ($Z_j^k = 0$)، آنگاه اطمینان حاصل گردد که در بازه زمانی بعد که ممکن است در بدترین حالت، پس از گذر H واحد زمانی باشد، آن وظیفه در بدترین حالت، در زمانی کمتر از تاریخ سررسید خود به اضافه زمان تاخیر اجرا گردد. تأخیر در بدترین حالت، برابر با صفر و در غیر این صورت برابر با $H + M_j - d_j$ است. محدودیت (p1l) برای تخصیص هزینه به وظایف استفاده می‌گردد. هزینه هر درخواست برابر است با هزینه ماشین مجازی انتخابی برای آن درخواست و مدت زمانی که از آن ماشین مجازی استفاده می‌گردد. محدودیت‌های (p1m) و (p1n) نشان می‌دهند که اولین درخواستی که به ازای هر صف k زودتر از بقیه به اتمام می‌رسد، از وظایفی باشد که بر روی همان صف در حال اجرا بوده‌اند و مهمتر از آن نشان می‌دهد که فقط یک درخواست می‌تواند چنین ویژگی داشته باشد. محدویت (p1o) نشان می‌دهد که مجموعه گره‌های انتخاب شده در هر صف، حداکثر به اندازه مینیمم تعداد وظایف یا مینیمم تعداد کل گره‌های موجود در آن صف باشد. محدویت‌های (p1p) تا (p1s) برای مدیریت متغیرهای زمانبندی استفاده می‌شوند و دامنه متغیرهای تصمیم را تعریف می‌کنند.

با توجه به غیرخطی بودن مسئله، قصد داریم با استفاده از روش [۵]، مدل خود را به یک مدل خطی تبدیل نماییم. بنابراین باید بخش غیرخطی تابع هدف را به صورت زیر تغییر دهیم. پس از تغییر تابع هدف و افزودن متغیرهای مصنوعی، محدودیت‌هایی به مدل اضافه می‌گردند که مدل جدید خطی شده به صورت مدل (p2) نشان داده شده است. بخش غیرخطی در مدل قبلی، با افزودن یک متغیر جدید ξ_n^k که نشانگر حداقل هزینه استقرار تمام درخواست‌ها است و شش محدودیت جدید جایگزین شده است. همچنین به ازای هر درخواست، یک متغیر غیرمنفی کمکی به صورت γ_{jn}^k که نشانگر هزینه اجرای درخواست j بر روی سرور n از صف k است، تعریف شده است.

$$min \sum_{j\epsilon J_k} \omega_j \left(\theta_j^k + \rho\widehat{\theta_j^k}\right) + \mu \sum_{\substack{n\epsilon N_k \\ v\epsilon V}} (s_v y_{nv}^k - \sum_{\substack{j\epsilon J_k \\ g\epsilon G_v}} g x_{jnvg}^k) + \sum_{\substack{j\epsilon J_k \\ n\epsilon N_k \\ v\epsilon V \\ g\epsilon G_v}} \frac{g}{s_v} c_v\, t_{jvg} x_{jnvg} + \sum_{n\epsilon N_k} \xi_n^k \quad \text{(p2a)}$$

Subject to:

(P1a) - (P1r) and:

$$\xi_n^k \geq \sum_{j\epsilon J_k} \gamma_{jn}^k ; \quad \forall n \in N_k \quad \text{(p2b)}$$

$$\gamma_{jn}^k \leq \pi_{jn}^k; \quad \forall j \in J_k, \quad \forall n \in N_k \quad \text{(p2c)}$$

$$\gamma_{jn}^k \leq \widehat{M_J}\alpha_{jn}^k ; \quad \forall j \in J_k, \quad \forall n \in N_k \quad \text{(p2d)}$$

$$\pi_{jn}^k - \widehat{M_J}\left(1 - \alpha_{jn}^k\right) \leq \gamma_{jn}^k; \quad \forall j \in J_k, \quad \forall n \in N_k \quad \text{(p2e)}$$

$$\xi_n^k \geq 0; \quad \forall n \in N_k \quad \text{(p2f)}$$

$$\pi_{jn}^k \geq 0; \quad \forall j \in J_k, \quad \forall n \in N_k \quad \text{(p2g)}$$

مطابق با محدودیت‌های (p2b) تا (p2g)، اگر $\alpha_{jn}^k = 1$ باشد، متغیر ξ_n^k برابر با هزینه اولین وظیفه‌ای خواهد بود که در صف k به اتمام می‌رسد. اگر:

- $\alpha_{jn}^k = 1$ باشد، بنابراین محدودیت (p2c) نسبت به محدودیت (p2d) (یعنی $\gamma_{jn}^k \leq \widehat{M_J}$) محدودکننده‌تر خواهد بود. زیرا می‌دانیم که $\pi_{jn}^k \leq \widehat{M_J}$. همچنین از محدودیت (p2e) نتیجه $\gamma_{jn}^k \geq \pi_{jn}^k$ حاصل می‌شود که در نهایت خواهیم داشت $\gamma_{jn}^k = \pi_{jn}^k$. محدودیت (p2b) نیز مجموع هزینه تمام وظایف را محاسبه کرده و در تابع هدف با هدف کمینه‌سازی استفاده می‌کند.
- اگر $\alpha_{jn}^k = 0$ باشد، محدودیت (p2d) باعث می‌گردد تا $\gamma_{jn}^k \leq 0$ گردد و با استفاده از محدودیت (p2g) مشخص می‌گردد که نهایتاً $\gamma_{jn}^k = 0$ است. محدویت (p2e) نیز برابر خواهد بود با $\gamma_{jn}^k \geq \pi_{jn}^k - \widehat{M_J}$، که همواره برقرار است.

بنابراین یک مسئله بهینه‌سازی از نوع کمینه‌سازی داریم. با توجه به اینکه تمام متغیرهای تصمیم مسئله (p1) و (p2) متغیرهای بولین هستند و متغیرهای θ_j^k ، $\widehat{\theta_j^k}$ و ξ_n^k متغیرهای وابسته و واقعی هستند. بنابراین مسئله دسته مسائل شبه بولی (Pseudo-Boolean) قرار داد. الگوریتم کلی استفاده شده برای توزیع درخواست‌ها و زمانبندی در الگوریتم ۱ نشان داده شده است.

الگوریتم ۱- مراحل روش توزیع و زمانبندی پیشنهادی

Distribution

Inputs: J, K
Outputs: J_k
1. Sort all jobs in central queue based on Earliest Deadline First (EDF)
2. Distribute all jobs based on sorted values as EDF/ EDF2/ EDF3 (see figure 2)
3. For each local queue J_k between $1, \dots, K$ do:
4. Divide jobs for each local queue J_k

Scheduling

Inputs: J_k, all info of jobs (deadline, tardiness, …) and all info of servers (cost, …)
Output: scheduling result
1. For each local queue J_k between $1, \dots, K$ do parallel:
2. Compute all predicted execution times t_{jnvg}
3. Set *P2* for all received jobs
4. Solve *P2* with *CPLEX* solver
5. Show scheduling results: for each job: 1) the best node, 2) the best VM type, and 3) the best number of GPUs.

همان‌طور که در الگوریتم ۱ نشان داده شده است، روش پیشنهادی دارای دو مرحله است. در مرحله اول، توزیع درخواست‌ها با یکی از سه روش پیشنهادی EDF، EDF2 و EDF3 (شکل ۲ را ببینید) انجام می‌گیرد. سپس با مشخص شدن وظایف مربوط به هر صف محلی، این صف‌های محلی به صورت کاملاً موازی با هم، وظایف خود را مطابق با مسئله (p2) زمانبندی می‌کنند.

۴- ارزیابی روش پیشنهادی

روش پیشنهادی به وسیله زبان پایتون و ابزار pyomo پیاده‌سازی شده است. تمام ویژگی‌های استفاده شده برای بخش ارزیابی در جدول ۲ نشان داده شده است. ارزیابی روش پیشنهادی در سه مجموعه آزمایش جداگانه برای بررسی زمان اجرای یک مجموعه وظایف، زمان اتمام زمانبندی یا همان زمان پاسخ و هزینه زمانبندی یک مجموعه وظایف انجام شده است.

جدول۲- ویژگی‌های استفاده شده برای ارزیابی روش پیشنهادی

ویژگی وظایف			
زمان ورود وظایف	تاریخ سررسید وظایف	وزن تأخیر وظایف	
توزیع پواسون با $\lambda = 5$	تصادفی	تصادفی	
ویژگی مدلسازی			
ابزار مدلسازی	ابزار solver خطی	شرط خاتمه MIPGap	
Pyomo	glpk	0.2	
ویژگی سرورها			
انواع ماشین‌های مجازی	انواع GPU	تعداد GPU	هزینه
Standard_NC6	K80	۱	۰/۵۶
Standard_NC12	K80	۲	۱/۱۳
Standard_NC24	K80	۴	۲/۲۵
Standard_NC24X	K80	۸	۴/۴۸
Standard_NV6	M60	۱	۰/۶۲
Standard_NV12	M60	۲	۱/۲۴
Standard_NV24	M60	۴	۲/۴۸
Standard_NV24X	M60	۸	۴/۹۶
In_house_server_1	Quadro P600	۸	۰/۴۴
In_house_server_2	Quadro P600	۲	۰/۱۱
In_house_server_3	GTX 1080Ti	۸	۱/۱۳

۴-۱- *بررسی زمان اجرای یک مجموعه وظایف*

به منظور بررسی زمان اجرای وظایف مختلف در روش پیشنهادی، الگوریتم زمانبندی در سه سناریو مختلف به صورت زیر اجرا گردیده است.

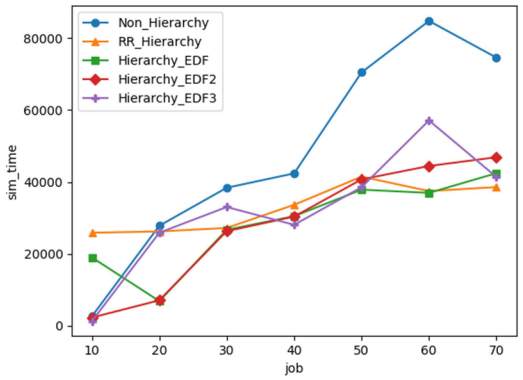
سناریو۱) توزیع درخواست با روش پیشنهادی EDF و زمانبندی با P2، که در نمودارها با نام Hierarchy_EDF نمایش داده شده است.

سناریو۲) توزیع درخواست با روش پیشنهادی EDF2 و زمانبندی با P2، که در نمودارها با نام Hierarchy_EDF2 نمایش داده شده است.

سناریو۳) توزیع درخواست با روش پیشنهادی EDF3 و زمانبندی با P2، که در نمودارها با نام Hierarchy_EDF3 نمایش داده شده است.

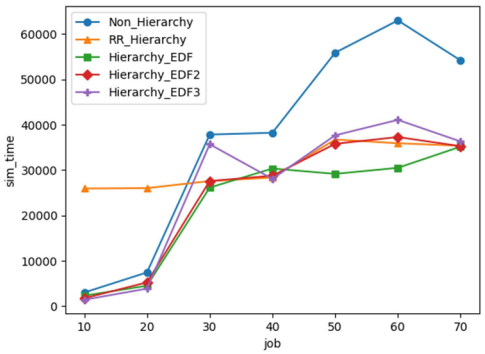
همچنین روش [۱۵] و [۵] که به ترتیب یک روش غیرسلسله‌مراتبی و سلسله مراتبی با توزیع درخواست نوبت گردشی هستند نیز پیاده‌سازی شده و در شرایطی مشابه با روش پیشنهادی مورد ارزیابی قرار گرفتند. نمودار مربوط به زمان اجرای وظایف (اتمام اجرای تمام وظایف محوله) در شرایط مشابه، برای روش پیشنهادی در سه سناریو مختلف و همچنین دو راهکار پیشین با نام Non_Hierarchy [۱۵] و با نام RR_Hierarchy [۵] در شکل ۳ نشان داده شده است.

نتایج، حاصل اجرای میانگین ۱۰ اجرای مختلف به روش سلسله‌مراتبی است و در هر اجرا تعداد وظایف از ۱ تا ۷۰ با گام ۱۰ تغییر یافته است. تعداد گره یا سرور برابر با |N|=3 و تعداد صف برابر با K=3 می‌باشد. محور عمودی نشانگر زمان اتمام اجرای تمام وظایف است.



شکل ۳- زمان اجرا بر تعداد وظایف، |N|=3 و K=3

مطابق آنچه در شکل ۳ نشان داده شده است، زمان اجرا در حالت غیرسلسله‌مراتبی، بسیار بیشتر از زمان اجرای سایر روش‌ها است. بدیهی است که چون در این حالت، فقط یک گره یا سرور مسئول زمانبندی تمام وظایف است، زمان اجرا افزایش یافته باشد. هر چند تمام سرورها در اختیار همان صف مرکزی هستند و در واقع زمانبندی به صورت کاملاً مرکزی انجام می‌گیرد، اما به دلیل زیاد شدن بار کاری همان صف مرکزی، امکان پاسخگویی محدود است و از یک مرحله‌ای به بعد که چندین درخواست در حال اجرا در سیستم وجود دارد و چندین درخواست جدید هم به سیستم اضافه شده است، امکان پاسخگویی وجود نخواهد داشت و ایراد اصلی چنین روشی، همین موردِ وجود یک نقطه شکست واحد (SPF)[۵۵۵] است. زمان اتمام اجرای روش پیشنهادی در هر سه سناریو تقریباً یکسان و کمتر از حالت غیرسلسله‌مراتبی هستند. روش RR_Hierarchy در تعداد وظایف کمتر دارای زمان اجرای قابل قبول است، اما با افزایش تعداد وظایف، زمان اجرای بالاتری از روش پیشنهادی که سلسله‌مراتبی است، خواهد داشت. زیرا قدرت روش پیشنهادی در تعداد وظایف بالاتر نمایان می‌گردد. از سه سناریو بررسی شده نیز در ابتدای نمودار، زمان اجرا تقریباً برابر است. اما با افزایش تعداد وظایف، روش Hierarchy_EDF که همان سناریو۱ است، عملکرد بهتری دارد، زیرا اکثر وظایفی را که دارای زمان اتمام یکسانی هستند را بین صف‌های مختلف توزیع می‌کند و هرگز اجازه نمی‌دهد، هیچ دو وظیفه‌ی متوالی به یک صف ارسال گردند. یک آزمایش مشابه با شکل ۳، با تعداد سرورهای برابر با |N|=5 نیز اجرا شده است که نتایج حاصل در شکل ۴ نشان داده شده است.

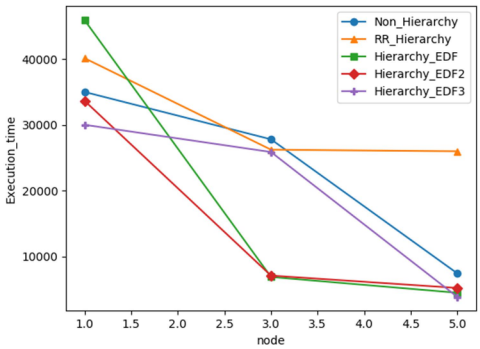


شکل ۴- زمان اجرا بر تعداد وظایف، |N|=5 و K=3

مطابق با آنچه که در شکل ۴ نشان داده شده است، بالا بودن زمان اجرا در روش غیرسلسله‌مراتبی کاملاً مشهود است. روش RR_Hierarchy در تعداد وظایف کمتر دارای زمان اجرای قابل قبولی است که با افزایش تعداد وظایف این زمان افزایش می‌یابد. اما روش پیشنهادی که حالت سلسله‌مراتبی دارد، در هر سه سناریو دارای زمان اجرای پایین‌تر است و مجدداً مطابق با نتایج بدست آمده، سناریو یک بهتر از سایرین است. زیرا اجازه نمی‌دهد دو وظیفه متوالی به یک صف ارسال گردند.

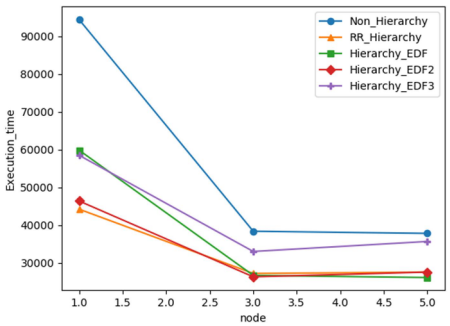
۴-۲- *بررسی زمان پاسخ*

این بخش زمان پاسخ روش پیشنهادی را در مقایسه با روشهای پیشین ارزیابی می‌کند. زمان پاسخ، نشانگر زمان اتمام شبیه‌سازی و تولید پاسخ زمانبندی برای یک مجموعه وظایف است. توجه داشته باشید که زمان اجرای وظایف که در بخش قبلی مورد بررسی قرار گرفت، نشانگر زمان اتمام اجرای تمام وظایف بود. اما این بار، هدف اندازه‌گیری زمان پاسخ روش زمانبندی پیشنهادی است که مجدداً در سه سناریو پیشین انجام شده است و نتایج در شکل‌های ۵ و ۶ نشان داده شده است. در تمام این نمودارها نیز میانگین ۱۰ اجرا گزارش شده است.



شکل ۵- زمان پاسخ بر تعداد سرورها، |J|=20 و K=3

مطابق آنچه در شکل ۵ نشان داده شده است، زمان پاسخ در حالت غیرسلسله‌مراتبی، بالاتر از سایر روش‌ها است. روش پیشنهادی در هر سه سناریو دارای زمان اجرای پایین‌تر، به خصوص با افزایش تعداد وظایف و گره‌های موجود است. اما همچنان سناریو اول دارای عملکرد بهتری است. آزمایشی مشابه در شکل ۶ نیز نشان داده شده است که دارای تعداد وظایف بیشتر و برابر با ۳۰ عدد وظیفه متفاوت است.



شکل ۶- زمان پاسخ بر تعداد سرورها، |J|=30 و K=3

نتایج شکل ۶ نیز نشانگر بالا بودن زمان پاسخ در حالت غیرسلسله‌مراتبی است. در روش پیشنهادی سلسله‌مراتبی، زمان پاسخ کمتر از سایر روش‌ها است و همچنان سناریو اول دارای عملکرد بهتری است.

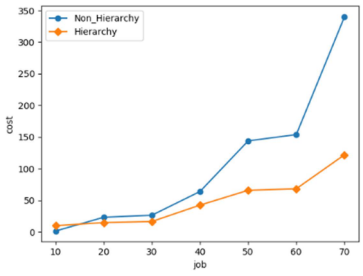
۴-۳- *بررسی هزینه*

به ازای هر زمانبندی انجام شده، هزینه‌ای مشخص می‌گردد که این هزینه به صورت معادله ۱ محاسبه گردیده است. این بخش به مقایسه هزینه زمانبندی در روش پیشنهادی و روش غیرسلسله‌مراتبی پرداخته است و نتایج مقایسه با افزایش تعداد وظایف محاسبه شده است.

$$cost_j = c_v t_{slot} \cdot \frac{g\, x_{jnvg}^k}{s_v} + \omega_j \theta_j^k \qquad (۱)$$

در معادله ۱، t_{slot} نشانگر بازه زمانی مربوط به هر اسلات است که با رخداد یکی از سه حالت (اتمام یک وظیفه، ورود یک وظیفه جدید و یا گذر یک مدت زمان معین) تمام شده و اسلات بعدی آغاز می‌گردد. در هر اسلات زمانی، فقط یک بار زمانبندی انجام می‌گیرد.

در این آزمایش، ابتدا مجموعه وظایف برای اندازه‌گیری هزینه مشخص گردیده است و سپس هزینه تمام اسلات‌های زمانی مورد نیاز برای تکمیل آن وظایف محاسبه شده و مجموع آنها در نمودارهای مربوطه ترسیم شده است. نمودار شکل ۷، مجموع هزینه را برای وظایف ۱۰ تا ۷۰ با گام افزایشی ۱۰ محاسبه کرده و نمایش داده است. در این نمودار، تعداد صف‌ها برابر با ۳ و تعداد سرور موجود در هر صف، سه سرور بوده است. اختلاف هزینه به حدود ۰/۳۸ دلار رسیده است و این هزینه در روش پیشنهادی کمتر از روش غیرسلسله‌مراتبی است.



شکل ۷- هزینه بر تعداد وظایف، |N|=1 و K=3

۵- نتیجه‌گیری و کارهای آتی

به دلیل استفاده از سرورهای ابری برای پاسخ به درخواست‌هایی که غالباً مربوط به مرحله آموزش شبکه‌های عصبی و کاربردهای یادگیری ماشین است، تخصیص منابع این سرورها بسیار چالش‌برانگیز است. مقاله حاضر ابتدا روشی برای توزیع درخواست‌ها و سپس روشی برای زمانبندی درخواست‌ها ارائه داد. روش پیشنهادی توانست با توزیع درخواست‌ها براساس تاریخ سررسید وظایف، بهترین توزیع را با کاهش زمان و هزینه انجام دهد. همچنین روش زمانبندی پیشنهادی نیز توانست نسبت به روش‌های پیشین، در مدت زمان کمتر و با استفاده از منابع کمتری به درخواست‌ها پاسخ دهد. قصد داریم به عنوان کارهای آتی، هزینه مهاجرت وظایف را در نظر گرفته و روش زمانبندی پیشنهادی را با هدف کاهش هزینه مهاجرت توسعه دهیم.

منابع و مراجع

[1] Peddie, Jon. "What is a GPU?" *In The History of the GPU-Steps to Invention*, pp. 333-345. Cham: Springer International Publishing, 2023.

[2] Buber, Ebubekir, and D. I. R. I. Banu. "Performance analysis and CPU vs GPU comparison for deep learning." *In 2018 6th International Conference on Control Engineering & Information Technology (CEIT)*, pp. 1-6. IEEE, 2018.

[3] Keckler, Stephen W., William J. Dally, Brucek Khailany, Michael Garland, and David Glasco. "GPUs and the future of parallel computing." *IEEE micro 31*, no. 5, pp. 7-17, 2011.

[4] Arunarani, A. R., Dhanabalachandran Manjula, and Vijayan Sugumaran. "Task scheduling techniques in cloud computing: A literature survey." *Future Generation Computer Systems 91*, pp. 407-415, 2019.

[5] Filippini, Federica, Marco Lattuada, Arezoo Jahani, Michele Ciavotta, Danilo Ardagna, and Edoardo Amaldi. "Hierarchical Scheduling in on-demand GPU-as-a-Service Systems." *In 2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, pp. 125-132. IEEE, 2020.

[6] J. Wu and B. Hong, "Collocating cpu-only jobs with gpuassisted jobs on gpu-assisted hpc," in *CCGrid, 2013 13th IEEE/ACM International Symposium on*, pp. 418–425, IEEE, 2013.

[7] O. Kayiran, N. C. Nachiappan, A. Jog, R. Ausavarungnirun, M. T. Kandemir, G. H. Loh, O. Mutlu, and C. R. Das, "Managing gpu concurrency in heterogeneous architectures," in *Microarchitecture, 47th Annual IEEE/ACM International Symposium on*, pp. 114–126, IEEE, 2014.

[8] C. Reano, F. Silla, D. S. Nikolopoulos, and B. Varghese, "Intranode memory safe gpu co-scheduling," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 5, pp. 1089–1102, 2018.

[9] S. Kato, K. Lakshmanan, R. Rajkumar, and Y. Ishikawa, "Timegraph: Gpu scheduling for real-time multi-tasking environments," in *Proc. USENIX ATC*, pp. 17–30, 2011.

[10] Y. Kang, W. Joo, S. Lee, and D. Shin, "Priority-driven spatial resource sharing scheduling for embedded graphics processing units," *Journal of Systems Architecture*, vol. 76, pp. 17–27, 2017.

[11] A.-M. Oprescu and T. Kielmann, "Bag-of-tasks scheduling under budget constraints," in *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pp. 351–359, IEEE, 2010.

[12] Z. Cai, X. Li, R. Ruiz, and Q. Li, "A delay-based dynamic scheduling algorithm for bag-of-task workflows with stochastic task execution times in clouds," *Future Generation Computer Systems*, vol. 71, pp. 57–72, 2017.

[13] Åsberg, Mikael, Thomas Nolte, Shinpei Kato, and Ragunathan Rajkumar. "Exsched: An external cpu scheduler framework for real-time systems." *In 2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pp. 240-249. IEEE, 2012.

[14] Ukidave, Yash, Xiangyu Li, and David Kaeli. "Mystic: Predictive scheduling for gpu based cloud servers using machine learning." *In 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 353-362. IEEE, 2016.

[15] Jahani, Arezoo, Marco Lattuada, Michele Ciavotta, Danilo Ardagna, Edoardo Amaldi, and Li Zhang. "Optimizing on-demand gpus in the cloud for deep learning applications training." *In 2019 4th International Conference on Computing, Communications and Security (ICCCS)*, pp. 1-8. IEEE, 2019.

[16] Lattuada, Marco, Eugenio Gianniti, Danilo Ardagna, and Li Zhang. "Performance prediction of deep learning applications training in GPU as a service systems." *Cluster Computing*, 1-24, 2022.

[] Round Robin

[] Earliest Deadline First

[] Single Point of Failure