فصلنامه مدیریت صنعتی دانشکده علوم انسانی، دانشگاه آزاد اسلامی
واحد سنندج – سال پانزدهم/ شماره ۵۲ / تابستان ۱۳۹۹

تخصیص ایستای وظایف در سیستم‌های توزیع‌شده با استفاده از الگوریتم ژنتیک موازی

منیره طاهری سروتمین
گروه مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد کرمان، کرمان، ایران
Email: mtaheri@iauk.ac.ir

تاریخ دریافت: ۹۹/۱۲/۰۳ * تاریخ پذیرش ۹۹/۰۶/۱۶

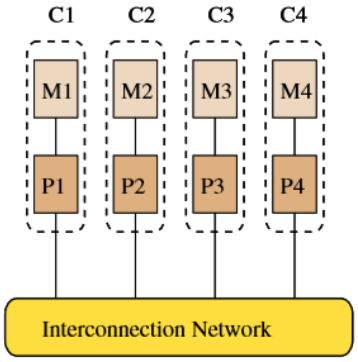
چکیده

در طی دو دهه اخیر، بالارفتن فوق‌العاده سرعت شبکه‌های رایانه ای و همچنین افزایش نیاز به سیستم‌هایی با کارایی بالا سبب شده است که محققان به پردازش‌های موازی و توزیع‌شده علاقه‌مند شوند. رشد سریع سیستم‌های توزیع‌شده باعث شده که مسائل گوناگونی در این زمینه مطرح شود. یکی از مهم‌ترین مسائلی که موردتوجه محققان زیادی قرارگرفته، مسئله تخصیص وظایف در این‌گونه محیط‌ها است که به‌منظور به دست آوردن بهره‌وری مؤثر از سیستم انجام می‌شود. مسئله تخصیص وظایف به‌جز در معدود موارد خاص جز مسائل NP–کامل است؛ بنابراین از فرایندهای اکتشافی برای دستیابی به راه‌حل‌های زیربهینه در مدت‌زمان مطلوب استفاده می‌شود. اگرچه از روش‌های مختلف در تحقیقات استفاده‌شده، اما هنوز پیدا کردن روش مؤثر و کارا برای این مشکل موردنیاز و مطلوب است. در این پژوهش از الگوریتم ژنتیک موازی برای پیدا کردن راه‌حل بهینه برای تخصیص یک گراف از وظایف به پردازنده‌ها در سیستم توزیع‌شده استفاده‌شده است. نتایج نشان داد الگوریتم پیشنهادی می‌تواند تخصیص‌های بهینه یا نزدیک بهینه برای مسائل با اندازه‌های گوناگون ارائه دهد. همچنین روش پیشنهادی توانست در زمان بسیار سریع‌تر از الگوریتم ژنتیک سنتی و با تسریع فراخطی، مسائل با اندازه‌های بزرگ و متوسط را حل کند.

کلمات کلیدی: سیستم توزیع‌شده، تخصیص وظایف، الگوریتم ژنتیک موازی، الگوریتم ژنتیک جزیره‌ای، الگوریتم ژنتیک سلولی.

۱-مقدمه

سیستم‌های توزیع‌شده[1] به‌عنوان یک بستر قدرتمند برای اجرای برنامه‌های موازی با کارایی بالا و به‌عنوان یک جایگزین بجای ماشین‌های موازی حجیم ظهور پیداکرده‌اند. یک سیستم توزیع‌شده شامل یک مجموعه از پردازنده‌های ناهمگن است از طریق یک شبکه ارتباطی به هم متصل شده‌اند(Akbari & Rashidi, 2016; Jiang, 2016; Neelakantan & Sreekanth, 2016; Wu, 2017). یک برنامه کاربردی موازی می‌تواند به تعدادی وظیفه تقسیم شود و به‌طور هم‌زمان روی پردازنده‌های مختلف سیستم اجرا شود؛ اما کارایی آن روی یک سیستم توزیع‌شده عمدتاً به تخصیص وظایف برنامه روی پردازنده‌های موجود در سیستم وابسته است. اگر تخصیص به‌درستی پیاده‌سازی نشود، پردازنده‌های سیستم ممکن است بجای انجام محاسبات مفید اکثر زمانشان را منتظر یکدیگر بمانند. یک تخصیص بهینه، تخصیصی است که بتواند هزینه‌های مربوط به سیستم را کمینه کند. شکل ۱ یک نمونه سیستم توزیع‌شده را نشان می‌دهد.



شکل شماره (۱): یک سیستم توزیع‌شده نمونه

در سیستم توزیع‌شده، بر اساس زمانی که در آن تصمیمات تخصیص گرفته می‌شوند، سیاست تخصیص ممکن است ایستا یا پویا باشد. در حالت عمومی، مسئله تخصیص وظایف در هر دو حالت، یک مسئله NP -کامل است. بعلت اهمیت کلیدی این مسئله در کارایی، مطالعات زیادی روی آن انجام‌شده و روش‌های متعددی در مقالات گزارش شده است(Aggarwal, Verma, & Singh, 2018a; Chaubey & Gupta, 2019; Hu, Li, & He). در میان کارهای انجام‌شده، کاهش زمان اجرا یا همان کاهش هزینه سیستم و قابلیت اطمینان دو مورد از مهم‌ترین اهداف برای بالا بردن کارایی این سیستم‌ها هستند(Attiya & Hamam, 2003b; R. Bharati, Shahakar, & Mahajan; Govil & Kumar, 2011).

راه‌حل‌های مختلفی پیشنهادشده‌اند تا مسئله تخصیص وظایف را حل کنند. آن‌ها به‌طورکلی به دودسته اصلی تقسیم می‌شوند، با نام‌های الگوریتم‌های دقیق و روش‌های اکتشافی. الگوریتم‌های دقیق ممکن است با استراتژی‌های مختلف ازجمله تئوری گراف(R. Bharati et al., 2014; R. D. Bharati, Jagtap, Gupta, & Landge, 2013) جستجوی فضای حالت (Govil, 2011; Khan & Govil, 2013b; Shatz, Wang, & Goto, 1992) و برنامه‌نویسی ریاضیاتی (Braun, Siegel, Maciejewski, & Hong, 2008; Srinivasan & Jha, 1999; Vidyarthi & Tripathi, 2001) انجام می‌شوند. در این الگوریتم‌ها زمان با اندازه مسئله (افزایش تعداد وظایف) به‌صورت نمایی رشد می‌کند، بنابراین آن‌ها درواقع برای مسائل کوچک به کار می‌روند. از سوی دیگر روش‌های اکتشافی ابزارهای سریع و قدرتمندی برای بدست آوردن راه‌حل‌های زیربهینه فراهم می‌کنند.

[1]. Distributed system

از الگوریتم‌های ژنتیک موازی یا PGA[۲] ها به‌طور موفق در محدوده وسیعی از مسائل بهینه‌سازی استفاده‌شده است(Augustine & Raj, 2016; Cai, Cai, Chandrasekaran, & Zheng, 2016; Feng et al., 2017; Zhang et al., 2016). قدرت خوب این الگوریتم‌ها در مسائل با پیچیدگی بالا منجر به افزایش تعداد کاربرد آن‌ها در رشته‌های هوش مصنوعی، بهینه‌سازی‌های عددی و ترکیبی، تجارت، مهندسی و غیره شده است به همین علت در این پژوهش از آن‌ها استفاده می‌شود (Alba & Troya, 1999). این الگوریتم‌ها علاوه بر کاهش زمان محاسبات، منجر به افزایش اکتشافات و تنوع بهتر در مقایسه با الگوریتم‌های ژنتیک ترتیبی می‌شوند(Kacprzyk & Pedrycz, 2015).

در این پژوهش از راه‌حل الگوریتم ژنتیک موازی برای نگاشت مجموعه از وظایف به منابع به‌صورت ایستا در محیط‌های محاسباتی توزیع‌شده ناهمگن به منظور کاهش زمان اجرا استفاده شده است. نتایج مربوط به کاهش زمان اجرای این مسئله با الگوریتم‌های ژنتیک موازی نسبت به الگوریتم ژنتیک ترتیبی و همچنین نتایج بررسی میزان تسریع و کارایی این الگوریتم نسبت به نسخه ترتیبی در ادامه قرار داده شده است. نتایج نشان داد که روش پیشنهادی توانست این مسئله را حل نموده و به تسریع فراخطی دست یابد. ویژگی ها و قابلیت های خوب الگوریتم های ژنتیک موازی باعث شد حتی حل مسئله مذکور با روش پیشنهادی تنها روی یک پردازنده و بدون موازی سازی سخت افزاری با سرعت بالاتری نسبت به نسخه استاندارد الگوریتم ژنتیک انجام شود که این نتایج به صورت مفصل در بخش مربوط به خود توضیح داده شده است.

این مقاله به بخش‌های ذیل سازمان‌دهی می‌شود: بخش ۲ به ارائه پیشینه تحقیق، تعریف مسئله، روش پیشنهادی می پردازد. در بخش ۴ نتایج تجربی آمده است و منابع در بخش ۴ ارائه‌شده است.

۲- روش شناسی پژوهش

الف) پیشینه نظری

سیستم‌های توزیع‌شده به طور وسیع برای اجرای کاربردهای محاسباتی حجیم با نیازهای محاسباتی گوناگون استفاده می‌شوند. در چنین سیستم‌هایی، یک برنامه موازی می‌تواند به یک تعداد از وظایف هماهنگ تقسیم شود که به پردازنده‌های مختلف برای اجرا توزیع می‌شوند؛ اما هزینه‌ای که باید در قبال این مزایا بایستی پرداخته شود، افزایش پیچیدگی نرم‌افزار، افت کارایی و کاهش سطح امنیت است. با وجود تمام این اشکالات هنوز هم علاقه زیادی به ساخت و نصب سیستم‌های توزیعی در سرتاسر جهان وجود دارد(Attiya & Hamam, 2006; Wu, 2017).

کارایی یک برنامه کاربردی موازی که روی یک سیستم توزیع‌شده اجرا می‌شود به تخصیص وظایف تقسیم شده به پردازنده‌های در دسترس در سیستم وابسته است. در یک سیستم توزیع‌شده، تخصیص نامناسب وظایف می‌تواند به کارگیری صحیح توان در این سیستمها را تحت تأثیر قرار دهد و سبب ایجاد سربار مفرط ارتباطی یا بهره وری کم منابع شود و در موازی سازی انحراف ایجاد نماید(Attiya & Hamam, 2006; Q. Kang, He, & Wei, 2013). تخصیص ظایف یک مسئله رایج در سیستم‌های توزیع‌شده می‌باشد. در نتیجه برای بهره برداری از قابلیت‌های سیستم محاسباتی توزیع‌شده و همچنین برای یک موازی سازی مؤثر، وظایف یک برنامه موازی باید به درستی به پردازنده‌های در دسترس سیستم اختصاص داده شود. بهترین تخصیص وظایف به بهترین توازن سیستم منجر می‌شود(Tyagi & Gupta, 2018).

در سیستم توزیع‌شده، بر اساس زمانی که در آن تصمیمات تخصیص گرفته می‌شوند، سیاست تخصیص ممکن است ایستا یا پویا باشد. در یک تخصیص وظیفه ایستا، فرض می‌شود که اطلاعات مربوط به وظیفه و خصوصیات پردازنده از پیش و قبل از اجرای وظایف در اختیار هستند. تخصیص ایستا هنگامی انجام می‌شود که وظایف در یک فاز برنامه ریزی افلاین نگاشت می‌شود، از قبیل برنامه ریزی تخصیص در محیط تولید برای روز بعد(Menghani, 2010). تکنیک‌های نگاشت ایستا یک مجموعه ثابت از کاربردها، یک مجموعه ثابت از ماشین‌ها، و یک مجموعه ثابت از برنامه‌های کاربردی و خصوصیات ماشین را به عنوان ورودی می‌گیرد و یک مجموعه تک و ثابت از نگاشت‌ها را تولید می‌کند(Khan & Govil, 2013a). نگاشت پویا هنگامی انجام

[2]. Parallel Genetic Algorithm

می‌شود که کارها در یک حالت بلادرنگ به صورت آنلاین نگاشت می‌شوند، مثلاً هنگامی که وظایف در بازه‌های زمانی نامشخص می‌رسند بلافاصله نگاشت می‌شوند(Hu, Li, & He, 2019).

۱. تعریف مسئله

مسئله در این پژوهش به یک تخصیص ایستای بهینه از وظایف یک برنامه کاربردی موازی روی پردازنده‌ها در سیستم توزیع‌شده مربوط است. یک سیستم توزیع‌شده از دو مجموعه ساخته شده است، مجموعه پردازنده‌های ناهمگن $P=(P_1, P_2, \ldots, P_n)$ که به‌وسیله خطوط ارتباطی به هم متصل شده‌اند و مجموعه $T=(t_1, t_2, \ldots, t_m)$ از وظایف برنامه که درمجموع یک هدف مشترک را شکل می‌دهند. هزینه اجرای یک وظیفه اجرایی روی پردازنده‌های مختلف متفاوت است و هزینه‌های اجرا به شکل یک ماتریس با ترتیب $m\times n$ و با نام ماتریس هزینه اجرا[۳] مشخص می‌شود. مشابه آن، هزینه ارتباط بین دو وظیفه به شکل یک ماتریس و با نام ماتریس هزینه ارتباطی بین وظیفه[۴] و با ترتیب $m\times m$ داده می‌شود.

در شکل ۲ یک سیستم توزیع‌شده با پردازنده‌های مختلف، گراف تعامل وظیفه (TIG)[۵] و جداول هزینه اجرا و هزینه ارتباط نمایش داده‌شده است. قسمت (الف) یک برنامه کاربردی موازی را به‌وسیله یک گراف تعامل وظیفه $G(V,E)$ نشان می‌دهد. در گراف تعامل وظیفه، رأس‌ها نشان‌دهنده وظایف و لبه‌ها نشان‌دهنده روابط بین وظایف است. در این گراف V مجموعه وظایف و E مجموعه لبه‌ها را مشخص می‌کند. هر وظیفه $i \in V$، بار پردازشی معین دارد. در قسمت (ب) هزینه‌ها یا همان زمان‌های اجرای وظایف روی سه پردازنده مختلف، با ماتریس هزینه های اجرا را نشان داده است.

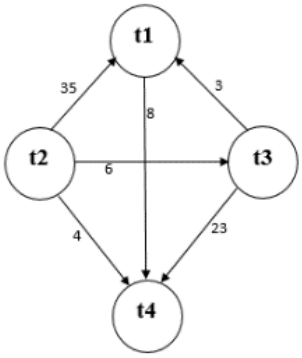
قسمت (ج) ماتریس هزینه‌های ارتباطی را نشان می‌دهد که این هزینه‌ها از روی گراف TIG بدست آمده است. به‌طور خلاصه در این مسئله یک مجموعه از M وظیفه وجود دارد که بیان می‌کنند یک برنامه کاربردی موازی باید روی یک سیستم توزیع‌شده N پردازنده‌ای اجرا شود.

ماتریس هزینه ارتباط

	t1	t2	t3	t4
t1	0	0	0	8
t2	35	0	6	4
t3	3	0	0	23
t4	0	0	0	0

ماتریس هزینه اجرا

	p_1	p_2	p_3
t_1	31	4	14
t_2	1	5	6
t_3	2	4	24
t_4	3	28	10



شکل شماره (۲): یک نمونه گراف تعامل وظیفه. (ب) ماتریس هزینه‌های اجرا. (ج) ماتریس هزینه‌های ارتباط

۲. فرموله سازی مسئله

فرموله کردن یک مدل ریاضیاتی برای مسئله تخصیص ایستای وظایف شامل دو گام است: ۱- فرموله کردن یک تابع هزینه برای نمایش هدف اصلی ۲- فرموله کردن برخی محدودیت‌ها یا نامساوی‌ها برحسب نیازهای وظایف و در دسترس بودن منابع سیستم. برای انجام آن X یک ماتریس دودویی MxN متناظر با تخصیص M وظیفه به N پردازنده است به‌گونه‌ای که:

$$X_{ip} = \begin{cases} 1 \\ 0 \end{cases}$$ رابطه (۱)

اگر وظیفه i به پردازنده p تخصیص داده شود، $X_{ip}=1$، در غیر این صورت، $X_{ip}=0$.

محدودیت مکان:

[3]. Execution Cost Matrix (ECM)
[4]. Inter Task Communication Cost Matrix (ITCCM)
[5]. Task Interaction Graph (TIG)

هر وظیفه باید فقط به یک پردازنده اختصاص یابد که به‌طور کامل و بدون قبضه شدن اجرا شود. رابطه زیر باید برای هر وظیفه حفظ شود.

$$\sum_p X_{ip} = 1$$ رابطه (۲)

تابع هزینه:

طبق فرمول ۳، تخصیص وظایف به پردازنده‌ها می‌تواند به‌صورت تابع X به شکل زیر تعریف شود:

$$X: T \rightarrow B \text{ such that } X(i) = k;$$ رابطه (۳)

اگر i مین وظیفه به k مین پردازنده اختصاص یابد.

هزینه اجرای پردازنده[۶]

هزینه اجرا وظیفه t_i، روی یک پردازنده P_k با ec_{ik} مشخص می‌شود که مقدار کل هزینه مورد نیاز برای اجرای t_i روی آن پردازنده در طول فرایند اجراست. اگر یک وظیفه قابل‌اجرا روی یک پردازنده خاص نیست، هزینه اجرا مربوطه بی‌نهایت (∞) در نظر گرفته می‌شود؛ بنابراین در یک تخصیص وظیفه X، هزینه اجرای پردازنده برای همه وظایف اختصاص داده شده به پردازنده k ام می‌تواند به‌صورت رابطه ۴ محاسبه شود(P. Yadav, M. Singh, & K. Sharma, 2011):

$$PEC(X)_k = \sum_{i=1}^{m} ec_{ik} x_{ik}$$ رابطه (۴)

هزینه ارتباط بین پردازنده[۷]

هزینه ارتباط بین پردازنده‌ها cc_{ij}، زمانی محاسبه می‌شود که بعلت ارتباط بین وظایف، داده از یک وظیفه به یک وظیفه دیگر انتقال داده شود و زمانی این هزینه در محاسبات اضافه می‌شود که وظایف t_i و t_j در پردازنده‌های جداگانه در طول فرآیند اجرا مستقر شوند. اگر دو وظیفه روی پردازنده یکسان اجرا شوند $cc_{ij}=0$ در نظر گرفته می‌شود؛ بنابراین در یک تخصیص وظیفه X، هزینه ارتباط بین پردازنده‌ها برای k مین پردازنده می‌تواند به‌صورت زیر محاسبه شود (P. Yadav et al., 2011):

$$IPCC(X)_K = \sum_{i=1}^{m} \sum_{j>i}^{m} (cc_{ij}) X_{ik} X_{jb}$$ رابطه (۵)

هزینه کل پردازنده k ام جمع هزینه اجرا پردازنده و هزینه ارتباط پردازنده برای k مین پردازنده تحت یک تخصیص وظیفه X است(P. Yadav et al., 2011):

$$T_{cost}(X)_k = PEC(X)_k + IPCC(X)_k$$ رابطه (۶)

و هزینه کل سیستم به شرح زیر محاسبه می‌شود(P. Yadav et al., 2011) :

$$S_{cost}(x) = \sum_{k=1}^{n} T_{cost}(x)_k$$ رابطه (۷)

مدل هزینه سیستم[۸]

با حساب کردن محدودیت‌های منابع سیستم، تخصیص وظیفه برای هزینه سیستم به‌صورت زیر فرموله می‌شود (P. Yadav et al., 2011):

$$min. S_{cost}(X)$$ رابطه (۸)

$$s.t. \sum_{k=1}^{n} X_{ik} = 1 \; \forall i = 1,2,3, \dots, m$$ رابطه (۹)

$$X_{ik} \in \{0,1\} \; \forall i, k.$$ رابطه (۱۰)

در این مدل محدودیت ۹ قرار داده‌شده است تا هر وظیفه دقیقاً به یک پردازنده اختصاص یابد و محدودیت ۱۰ تضمین می‌کند که X_{ik} یک متغیر تصمیم است.

ب) پیشینه تجربی

6. Processor Execution Cost (PEC)
7. Inter Processor Communication Cost (IPCC)
8. System Cost Model

بعلت طبیعت پیچیده مسئله تخصیص وظایف، تکنیک‌های مؤثر جدید همیشه برای بدست آوردن بهترین راه حل ممکن در زمان محاسباتی قابل قبول مطلوب هستند. مسئله تخصیص وظیفه به طور وسیع مطالعه شده است و روشهای متعددی در مقالات گزارش شده‌اند (Jiang, 2016; P. K. Yadav, M. Singh, & K. Sharma, 2011). مسئله مورد تحقیق برای اولین بار توسط استون[۹] معرفی شد(Stone, 1977). کار اصلی استون وضع کردن مدل TIG برای شرح وظایف اجرایی بود. در مدل معرفی شده در هر زمان دقیقا یک وظیفه روی یک پردازنده اجرا می شد.

در یک مطالعه، آگاروال[۱۰] و همکاران یک روش کارآمد برای تخصیص منابع با استفاده از برنامه ریزی ترکیبی و بهینه سازی در سیستم توزیع شده ارائه دادند. آن ها در این تحقیق با ارائه راهکار زمانبندی موثر و بهینه سازی مدیریت منابع با استفاده از بهینه سازی کلونی مورچه ها و زمانبندی راندرابین[۱۱] کار کرده اند تا به فواصل اجرای کم با احتمال نرخ خطای کمتر دست یابند(Aggarwal, Verma, & Singh, 2018b).

جیانگ[۱۲] در مقاله اش با عنوان بررسی تخصیص وظایف و توازن بار در سیستم های توزیع شده، به بررسی ویژگی های کلی سیستم های توزیع شده پرداخته است. او در مقاله اش، مطالعات مربوط به تخصیص وظیفه و توازن بار را با توجه به جنبه های مختلفی بررسی کرده و در مورد مسیرهای تحقیقاتی آینده بحث کرده است(Jiang, 2015). در مقاله دیگری با عنوان یک الگوریتم زمانبندی کار اصلاح شده برای سیستم های توزیع ناهمگن با محدودیت مصرف انرژی، هو[۱۳] و همکاران، برنامه زمانبندی وظایف را پیشنهاد داده اند تا ضمن به ارضاء محدودیتهای انرژی در سیستم های توزیع ناهمگن ، طول زمانبندی برنامه های موازی را به حداقل برسانند. نتایج آزمایشی نشان می دهد که در مقایسه با الگوریتم های موجود، الگوریتم جدید می تواند در شرایط محدودیت مصرف انرژی به طول برنامه ریزی بهتر برسد(Hu et al., 2019).

اکبری و رشیدی یک الگوریتم زمانبندی چند منظوره بر اساس بهینه سازی فاخته برای مسئله تخصیص وظیفه در زمان کامپایل در سیستم های ناهمگن پیشنهاد داده اند. این الگوریتم مبتنی بر الگوریتم بهینه سازی فاخته زمانبندی چند منظوره است. آن ها در روش شان، اجتناب از بهینه محلی را تضمین می کنند. الگوریتم پیشنهادی آن ها در حالی که امکان جستجوی سراسری را در فضای مسئله برای سرعت بخشیدن به یافتن بهینه سراسری را فراهم می کند و یک زمانبندی نسبتاً بهینه با کمترین تعداد تکرار فراهم می کند(Akbari & Rashidi, 2016).

یاداو و همکاران[۱۴] یک روش اکتشافی برای تخصیص بهینه وظایف بمنظور آنالیز هزینه سیستم در سیستم‌های محاسباتی توزیع شده ناهمگن ارائه داده‌اند. در این مقاله، یک الگوریتم ارائه شده است که تلاش می‌کند وظایف را به پردازنده‌ها به صورت یکی یکی و بر پایه جمع خط ارتباطی[۱۵]، تخصیص دهد. این نوع سیاست، ارتباط بین پردازنده[۱۶] را کاهش می‌دهد و بنابراین هزینه سیستم را کمینه می‌کند (P. K. Yadav et al., 2011).

حامد[۱۷] در مقاله‌اش،یک الگوریتم ژنتیک برای بدست آوردن راه حل بهینه برای تخصیص وظایف در سیستم های توزیع شده ناهمگن ارائه داده است به گونه ای که بار اختصاص یافته به هر پردازنده متعادل باشد. نتایج تجربی اثربخشی این الگوریتم را نسبت به الگوریتم های معمولی نشان می دهد(Hamed, 2012) . در مقاله جدیدی با عنوان طراحی یک چارچوب تخصیص

---

9. Stone
10. Mr. Anuj Aggarwal
11. Round Robin
12. Yichuan Jiang
13. Yikun Hu
14. P. K.Yadav, M.P.Singh, Kuldeep Sharma
15. Communication Link Sum (CLS)
16. Inter Process Communication (IPC)
17. A.Y.Hamed

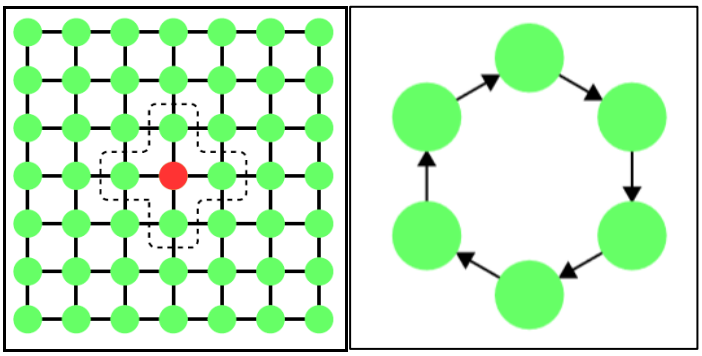
دهنده وظایف برای سیستم های توزیع شده، چابی[۱۸] و همکاران تلاش کرده اند تا یک چارچوب تخصیص دهنده وظیفه[۱۹] طراحی کنند تا بتواند الگوریتم های مختلف تخصیص وظیفه را پیاده سازی کند. با استفاده از این چارچوب تخصیص دهنده وظیفه می تواند در مواقع لازم مکانیزم تخصیص وظیفه را با یک مکانیزم جدید جایگزین کند.(Chaubey & Gupta, 2019).

در مطالعه دیگری نیلاکنتان و سریکان[۲۰] با عنوان تخصیص وظایف در سیستم های توزیع شده، با استفاده از تکنیک متعادل سازی بار ، زمان پاسخگویی به کارها با انتقال وظایف از رایانه های بارگذاری شده به رایانه های تحت بار به حداقل می رسد. این مقاله از تکنیک انتشار برای تعادل بین کارها در رایانه در یک سیستم توزیع شده استفاده می کند (Neelakantan & Sreekanth, 2016).

ج) مدل مفهومی

۱. روش پیشنهادی

روش پیشنهادی در این مقاله استفاده از الگوریتم‌های ژنتیک موازی است. الگوریتم‌های ژنتیک موازی به‌طورکلی شامل الگوریتم‌های ژنتیک مختلفی است که هرکدام از آن‌ها یک بخش از جمعیت یا جمعیت‌های مستقلی را با یا بدون ارتباط بین آن‌ها پردازش می‌کنند؛ بنابراین الگوریتم‌های ژنتیک موازی می‌توانند تنوع جمعیت را افزایش داده و زمان محاسباتی را کاهش دهند. این الگوریتم‌ها یک نوع جدید از الگوریتم‌های فراکتشافی را ارائه می‌دهند که کارایی و تأثیر بالاتری بر روی جمعیت ساختاری و اجرای موازی دارند(Alba & Troya, 1999). این الگوریتم‌ها می‌توانند به دو نوع اصلی تقسیم شوند، که نام آن‌ها ، نوع دانه‌درشت[۲۱] یا الگوریتم‌های ژنتیک جزیره‌ای و نوع ریزدانه[۲۲] یا الگوریتم‌های ژنتیک سلولی است. شکل ۳ (الف) طرح یک الگوریتم ژنتیک جزیره‌ای و (ب) طرح یک الگوریتم ژنتیک سلولی را نشان می دهد.



شکل شماره (۳): طرح یک الگوریتم ژنتیک جزیره‌ای با یک توپولوژی نمونه. طرح یک الگوریتم ژنتیک سلولی روی گراف توری ۷×۷. خط چین‌ها به همسایه‌های سلول مشخص شده اشاره می‌کنند.

برای موازی سازی الگوریتم ژنتیک کافی است به هر پردازنده اجازه دهیم تا الگوریتم ژنتیک ترتیبی خودش را در زیر جمعیت خودش اجرا کنند اما هر کدام تلاش کنند تا همان تابع را بیشینه کنند. در مدل‌های جزیره‌ای جمعیتِ هر اجرا به عنوان یک جزیره در نظر گرفته می‌شود. اغلب از جزایر به عنوان زیر جمعیت‌ها نام برده می‌شود که با هم جمعیت کل مدل جزیره را می‌سازند. اکثر مواقع جزایر به صورت مستقل در مدل اجرا مستقل کار می‌کنند؛ اما به صورت دوره‌ای راه حل‌ها بین جزایر مبادله می‌شوند که به این فرایند مهاجرت گفته می‌شود. هدف از داشتن یک توپولوژی، مهاجرت است. یک گراف جهت دار، جزایر را به عنوان گره‌هایش در نظر می‌گیرد و لبه‌های جهت دار، دو جزیره را به هم متصل می‌کند. در یک زمان مشخص اشخاص از هر جزیره به جزایر همسایه فرستاده می‌شوند، منظور از همسایه، جزایری است که می‌توانند از طریق لبه‌های جهت دار در توپولوژی به هم متصل شوند. افراد فرستاده شده مهاجر خوانده می‌شوند و آن‌ها بعد از یک فرایند انتخاب ثانویه در جزایر مقصد جای داده

---

18. Chaubey
19. Task allocator framework
20. Neelakantan, Sreekanth
21. Coarse-grain
22. Fine-grain

می‌شوند. در این روش جزایر می‌توانند با یکدیگر ارتباط برقرار کنند و رقابت کنند. جزایری که در نواحی با برازندگی پایین در فضای جستجو گیر می افتند می‌توانند بوسیله افرادی که از جزایر موفق‌تر می‌آیند از این مشکل بیرون آیند. . جدول ۱ شبه کد الگوریتم ژنتیک جزیره‌ای را مشخص می‌کند. این شبه کد، طرح کلی یک مدل جزیره‌ای ابتدایی را نشان می‌دهد. طرح‌های انتخابی زیادی وجود دارند که روی رفتار یک مدل جزیره‌ای تأثیر می‌گذارند.

جدول شماره (۱): شبه کد الگوریتم مدل جزیره‌ای با فاصله مهاجرت τ (Sudholt, 2015)

الگوریتم ۱. طرح یک مدل جزیره‌ای با فاصله مهاجرت τ
1: Initialize a population made up of subpopulations or islands, P(0) = (P1(0),…, Pm(0)). 2. Let t:=1. 3: loop 4: for each island i do in parallel 5: if t mod τ = 0 then 6: Send selected individuals from island Pi(t) to selected neighboring islands. 7: Receive immigrants Ii(t) from islands for which island Pi(t) is a neighbor. 8: Replace Pi(t) by a subpopulation resulting from a selection among Pi(t) and Ii(t) 9: end if 10: Produce Pi(t+1) by applying reproduction operators and selection to Pi(t). 11: end for 12: Let t:= t + 1. 13: end loop

برجسته‌ترین خصوصیت ژنتیک سلولی این است که هر جزیره فقط یک شخص را در بر دارد. در این مفهوم جزایر اغلب سلول‌ها خوانده می‌شوند که اصطلاح الگوریتم ژنتیک سلولی را شرح می‌دهند. به هر شخص فقط اجازه داده می‌شود که با همسایگانش ترکیب شود. این مدل از تعامل در هر نسل اتفاق می افتد. الگوریتم ژنتیک سلولی یک سیستم دانه–ریزتر را نتیجه می‌دهد. الگوریتم‌های ژنتیک سلولی مدل‌های ریزدانه‌ای هستند که مدل همسایگی، یا مدل‌های انتشار[۲۳] خوانده می‌شوند. تفاوت آن‌ها با مدل‌های جزیره‌ای این است که هیچ تکاملی در خود این سلول اتفاق نمی‌افتد یعنی هیچ تکاملی درون جزیره وجود ندارد. بهبودها فقط می‌توانند بوسیله سلول‌هایی بدست آیند که با یکدیگر تعامل دارند. جدول ۲ شبه کد الگوریتم ژنتیک سلولی را نشان می‌دهد.

جدول شماره (۲): شبه کد الگوریتم‌های ژنتیک سلولی (Sudholt, 2015)

الگوریتم ۲. طرح یک الگوریتم ژنتیک سلولی
1: Initialize all cells to form a population P(0) = (P1(0),...,Pm(0)). Let t:= 0. 2: loop 3: for each cell i do in parallel 4: Select a set Si of individuals from Pi(t) out of all cells neighboring to cell i. 5: Create a set Ri by applying reproduction operators to Si. 6: Create Pi(t+1) by selecting an individual from (Pi(t) ∪ Ri. 7: end for 8: Let t:= t + 1. 9: end loop

۲. نمایش ژنوتایپ مسئله

[23]. Diffusion model

یکی از موارد کلیدی در طراحی یک الگوریتم ژنتیک موازی، نمایش ژنوتایپ آن است، یعنی پیدا کردن یک نگاشت مناسب بین راه‌حل مسئله و ژنوتایپ آن(Augustine & Raj, 2016) . در مورد الگوریتم ژنتیک موازی، هر ژنوتایپ با یک راه‌حل کاندید برای مسئله متناظر است؛ بنابراین اجازه داده می‌شود که هر ژنوتایپ یک تخصیص وظیفه را با استفاده از بردار عددی M عنصری نمایش دهد که در آن موقعیت i اشاره دارد به شماره پردازنده‌ای که وظیفه شماره i به آن اختصاص داده می‌شود .Genotype[i]=p, pϵ(1,2,…,n)

شکل ۴ یک مثال تصویری برای ژنوتایپ را نشان می‌دهد و به یک تخصیص وظیفه اشاره دارد که پنج وظیفه را به سه پردازنده اختصاص می‌دهد. برای مثال Genotype[1]=2 به این معنی است که وظیفه ۱ به پردازنده ۲ اختصاص داده‌شده است و به همین ترتیب برای سایر موارد ادامه پیدا می‌کند.

وظیفه	1	2	3	4	5
پردازنده	2	1	3	3	1

شکل شماره (۴): یک مثال از ژنوتایپ مسئله

۳. دستورالعمل انتخاب

بخش اصلی فرایند انتخاب، انتخاب یک نسل برای ایجاد نسل بعدی به‌صورت تصادفی است. مناسب‌ترین افراد شانس بالاتری نسبت به افراد ضعیف‌تر دارند. در اینجا از روش انتخاب چرخ رولت برای انتخاب کروموزوم استفاده‌شده است.

۴. عملگر تقاطع

عملگر تقاطع با یک نرخ احتمال P_c مشخص می‌شود. در اینجا از روشی استفاده‌شده است که تلفیقی از روش‌های تقاطع تک نقطه‌ای، تقاطع دونقطه‌ای و تقاطع یکنواخت است. روش‌های تقاطع با احتمالات مختلف قرار داده شده‌اند. مقدار احتمال برای تقاطع تک نقطه‌ای $P_{singlePoint}=0.1$، احتمال تقاطع دونقطه‌ای $P_{doublePoint}=0.2$ و احتمال تقاطع یکنواخت $P_{uniform}=1-P_{singlePoint}-P_{doublePoint}$ در نظر گرفته شده است. هر بار که تابع تقاطع فراخوانی می‌شود با استفاده از روش چرخ رولت یکی از سه روش بالا برای انجام عمل تقاطع انتخاب می‌شود.

۵. عملگر جهش

همانند عملگر تقاطع، یک عملگر جهش با نرخ P_m برای تعیین اینکه چه تعداد از جمعیت بایستی دچار جهش شوند، در نظر گرفته شده است. همچنین m_u تعداد ژن‌هایی که در کروموزم بایستی با جهش تغییر پیدا را مشخص می‌کند. در اینجا مکانیزم جهش، پردازنده انتخابی از وظیفه منتخب به پردازنده‌ای دیگر که به‌طور تصادفی انتخاب شده است را تغییر می‌دهد.

۶. استراتژی‌های مربوط به الگوریتم ژنتیک جزیره‌ای

الگوریتم ژنتیک جزیره‌ای چندین زیرجمعیت (جزیره) را حالت موازی اجرا می‌کند. تبادل اطلاعات بین این زیرجمعیت‌ها در فواصل زمانی مشخص (دوره‌ای[۲۴]) در تکرار حلقه‌ها انجام می‌شود. با تبادل کروموزوم‌های برجسته[۲۵] بین زیرمجموعه‌ها، فضای جستجو زیرمجموعه‌ها دارای تنوع می‌شود تا به صورت مؤثر از همگرایی زودرس جلوگیری کند.

در اینجا از توپولوژی حلقوی برای مهاجرت استفاده‌شده و جهت آن یک‌طرفه است و با یک فرکانس مهاجرت، تعداد مشخص از بهترین افراد در زیرجمعیت‌ها برای مهاجرت انتخاب می‌شوند و یک کپی از این افراد به همسایه‌ها فرستاده می‌شوند. مهاجران رسیده در جزایر مقصد جایگزین بدترین افراد جامعه می‌شوند.

۷. شرط توقف

شرط توقف رسیدن به تعداد معین از تکرار نسل‌ها در نظر گرفته شده است. برای اندازه‌های مختلف مسئله این عدد تغییر می‌کند اما به‌گونه‌ای انتخاب شده است که این اطمینان وجود داشته باشد که این الگوریتم همگرا شود و الگوریتم بتواند پاسخ‌های این مسئله را پیدا کند.

۸. پیاده سازی

---

[24].epoch
[25].Outstanding chromosome

برای اجرا و پیاده‌سازی روش پیشنهادی از نرم‌افزار MATLAB R2012a استفاده‌شده و روی سیستمی با CPU=Core i3 و با سرعت 2.00GHz و RAM=4.00GB اجرا شده است. برای ارزیابی، دو پیکربندی سیستم در نظر گرفته شده است: یک سیستم توزیع‌شده ۴ کامپیوتری و یک سیستم توزیع‌شده ۶ کامپیوتری. به‌علاوه چهار اندازه جمعیت تصادفی به مقادیر M=8,12,16,20 استفاده‌شده است.

۹. اندازه‌گیری کارایی

همان‌طور که Alba (Alba, 2002) اشاره کرده است، مقایسه کارایی الگوریتم‌های تکاملی موازی و ترتیبیترتیبی فقط در موردی که آن‌ها به‌دقت یکسانی می‌رسند مفهوم دارد. در یک تقسیم‌بندی انجام‌شده توسط Alba (Alba, 2002) برای مقایسه الگوریتم ژنتیک موازی با الگوریتم ترتیبیترتیبی آن به دو نوع کلی تسریع قوی[۲۶] و تسریع ضعیف[۲۷] اشاره کرده است. تسریع قوی معمولاً استفاده نمی‌شود. در تسریع ضعیف زمان اجرای موازی یک الگوریتم با زمان اجرای ترتیبی آن مقایسه می‌شود و شامل دو مورد است:

تک ماشین/panmixia: الگوریتم موازی با نسخه استاندارد (panmictic) آن مقایسه می‌شود که روی یک ماشین تک اجرا می‌شود. برای نمونه، ممکن است یک مدل جزیره‌ای با m جزیره را با یک الگوریتم ژنتیک که یک جزیره را اجرا می‌کند مقایسه کند و به‌موجب آن، الگوریتمی که روی همه جزایر اجرا می‌شود در هر دو مورد یکی است.

رسمی[۲۸]: الگوریتم موازی که روی m ماشین اجرا می‌شود با همان الگوریتم که روی یک ماشین اجرا می‌شود مقایسه می‌شود.

۱۰. موازی‌سازی با جعبه‌ابزار محاسبات موازی متلب

برای هدف موازی‌سازی، جعبه‌ابزار محاسبات موازی (PCT[۲۹]) متلب انتخاب شده است. جعبه‌ابزار محاسبات موازی این امکان را فراهم می‌کند تا مسائل حجیم و محاسباتی را با استفاده از پردازنده‌های چندهسته‌ای، GPUها و کلاسترهای کامپیوتری اجرا شود. جعبه‌ابزار همچنین اجازه استفاده از همه قدرت پردازشی یک کامپیوتر رومیزی چندهسته‌ای را می‌دهد. استفاده کامل از پردازنده چندهسته‌ای یک کامپیوتر رومیزی، از طریق کارگرهایی امکان‌پذیر است که به‌طور محلی اجرا می‌شوند(Sharma & Martin, 2009).

این جعبه‌ابزار یک کلاستر محلی از کارگرها را برای ماشین محلی فراهم می‌کند و برنامه‌های کاربردی را روی کارگرهای محلی (موتورهای محاسباتی متلب) اجرا می‌کند. می‌توان همان برنامه کاربردی را بدون تغییر کد، روی یک کلاستر از کامپیوترها یا یک سرویس محاسباتی گرید، با استفاده از سرویس‌دهنده محاسبه توزیع‌شده متلب اجرا کرد. تعیین تعداد کارگرها در این جعبه‌ابزار به سخت‌افزارها و نحوه اتصال آن‌ها بستگی دارد. درهرصورت برای استفاده از پردازشگر یک سیستم چندهسته‌ای، معمولاً تعداد کارگرها برابر با تعداد هسته‌ها در نظر گرفته می‌شود. در نسخه ۲۰۱۲ متلب، جعبه‌ابزار به کاربران اجازه می‌دهد تا حداکثر ۱۲ کارگر متلب روی یک تک ماشین کار کنند.

۱۱. مجموعه داده[۳۰]

چون عموماً هیچ مجموعه داده پایه استاندارد پذیرفته شده برای ارزیابی الگوریتم تخصیص وظایف با هدف کاهش هزینه سیستم در سیستم‌های محاسباتی توزیع‌شده وجود ندارد، در این پژوهش برای آزمایش الگوریتم‌های ژنتیک موازی به‌وسیله شبیه‌سازی، محدوده‌ای از نمونه‌های مشابه که سایر پژوهشگران (Attiya & Hamam, 2003a, 2006; Q.-M. Kang, He, Song, & Deng, 2010) از آن‌ها استفاده کرده‌اند را تولید کردیم. یک مجموعه داده شبیه‌سازی بزرگ به‌وسیله یک مجموعه از پارامترهای مختلف ایجاد می‌شود که خصوصیات نمونه‌های مسئله را تعیین می‌کند پارامترهای اصلی در زیر توضیح داده شده‌اند.

---

[26]. Strong speedup
[27]. Weak speedup
[28]. orthodox
[29].Parallel Computing Toolbox
[30]. Dataset

- تعداد وظایف در یک برنامه TIG (M)
- تعداد پردازنده‌ها در یک سیستم محاسباتی توزیع‌شده (N)
- هزینه‌های اجرای برنامه روی پردازنده‌های مختلف
- هزینه‌های ارتباط بین وظایف

در این پژوهش، مقدار N به معنای تعداد پردازنده‌های سیستم‌های محاسباتی توزیع‌شده بین ۴ تا ۶ متغیر است. تعداد وظایف M با مقادیر ۸،۱۲،۱۶ و ۲۰ متفاوت است تا الگوریتم با مقیاس‌های مختلف وارسی شود.

۳-نتایج و بحث

الف) ارزیابی عملکرد روش پیشنهادی بر اساس روش مقایسه رسمی

با استفاده از روش مقایسه رسمی الگوریتم‌های ژنتیک موازی برای نمونه مسئله‌های تولید شده تصادفی تخصیص ایستا وظایف به پردازنده‌ها، یک بار به‌صورت موازی روی دو هسته پردازشگر و بار دیگر به‌صورت ترتیبی و روی یک هسته اجرا شدند و زمان اجرای آن‌ها اندازه‌گیری شد. مقادیر تسریع و کارایی این الگوریتم‌ها نسبت به نسخه ترتیبی خودشان محاسبه شد. جدول ۳ زمان اجرای الگوریتم‌های ژنتیک و جزیره‌ای و سلولی را برای اندازه‌های مختلف مسئله را نشان می‌دهد. لازم به ذکر است که نتایج در بخش الف جدول برای سیستم توزیع‌شده با ۴ و بخش ب جدول برای سیستم توزیع‌شده با ۶ کامپیوتر را نشان می‌دهد.

جدول شماره (۳): زمان اجرای مسئله با استفاده از الگوریتم‌های مختلف بر حسب ثانیه. (الف) سیستم توزیع‌شده با ۴ کامپیوتر. (ب) سیستم توزیع‌شده با ۶ کامپیوتر

(M,N)	T_{SIGA}	T_{PIGA}	T_{SCGA}	T_{PCGA}
(۸،۶)	۱۲/۰۳	۶/۸۴	۴۲/۴۰	۲۳/۴۵
(۱۲،۶)	۱۳/۴۳	۷/۳۴	۴۴/۸۶	۲۴/۵۹
(۱۶،۶)	۲۷/۰۳	۱۴/۶۱	۶۳/۰۵	۳۴/۴۸
(۲۰،۶)	۲۸/۱۱	۱۴/۷۹	۶۴/۰۸	۳۵/۰۴

(M,N)	T_{SIGA}	T_{PIGA}	T_{SCGA}	T_{PCGA}
(۸،۴)	۰/۹۲	۰/۸۱	۱/۹۹	۱/۳۶
(۱۲،۴)	۱/۱۹	۰/۸۸	۲/۲۳	۱/۳۹
(۱۶،۴)	۳/۷۱	۲/۱۸	۱۱/۱۸	۶/۴۷
(۲۰،۴)	۲۸/۸۹	۱۶/۰۵	۶۳/۵۹	۳۴/۸۹

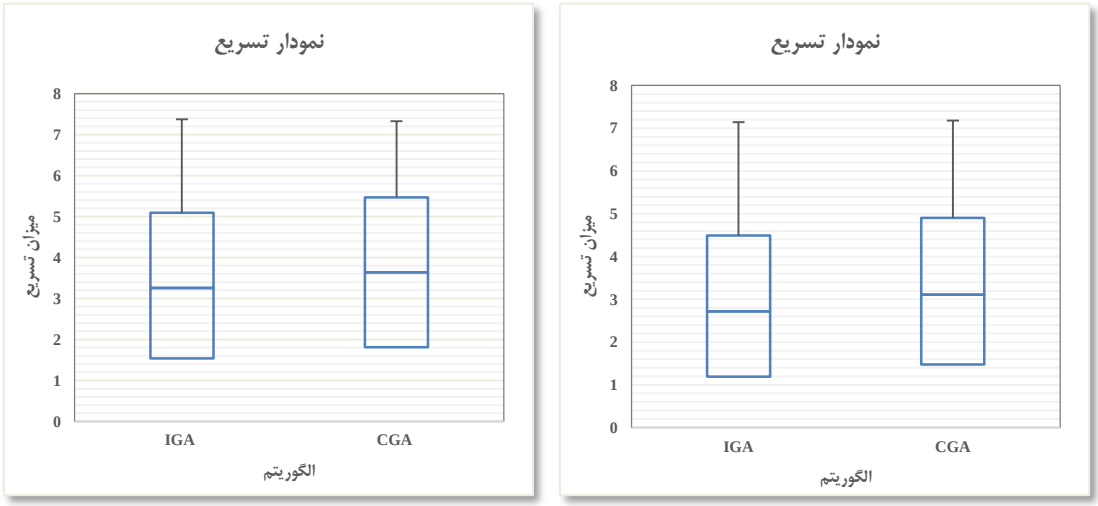
T_{SIGA}: زمان اجرا الگوریتم ژنتیک جزیره‌ای روی یک هسته پردازنده

T_{PIGA}: زمان اجرا الگوریتم ژنتیک جزیره‌ای روی دو هسته پردازنده به‌صورت موازی

T_{SCGA}: زمان اجرای الگوریتم ژنتیک سلولی روی یک هسته پردازنده

T_{PCGA}: زمان اجرای الگوریتم ژنتیک سلولی روی دو هسته پردازنده به‌صورت موازی

همان‌طور که در جدول۲ مشخص است هر دو الگوریتم ژنتیک جزیره‌ای و ژنتیک سلولی قادر هستند مسئله تخصیص ایستای وظایف به پردازنده‌ها در سیستم‌های توزیع‌شده را در زمان بسیار کمتری نسبت به نسخه ترتیبی اجرا کنند. ضمناً مقادیر جدول‌ها نشان می‌دهد که کارایی این الگوریتم‌ها با بالا رفتن اندازه مسئله افزایش پیدا می‌کند و نشان می‌دهد که الگوریتم‌های ژنتیک موازی برای مسائل حجیم و زمان‌بر، عملکرد بهتری دارند. نمودارهای شکل ۵ میزان تسریع الگوریتم‌های ژنتیک موازی را برای اندازه‌های مختلف مسئله نمایش می‌دهد. همان‌طور که از شکل ۵ پیداست با بالا رفتن اندازه مسئله میزان تسریع الگوریتم‌های ژنتیک موازی افزایش پیدا می‌کند.



شکل شماره(۵): نمودار میزان تسریع الگوریتم‌های ژنتیک جزیره‌ای و سلولی برای مسئله تخصیص ایستای وظایف به پردازنده‌ها با روش مقایسه رسمی. (الف) در یک سیستم توزیع‌شده با ۴ کامپیوتر. (ب) در یک سیستم توزیع‌شده با ۶ کامپیوتر

چندین فاکتور در ناقص بودن تسریع (مقدار تسریع کمتر از دو پردازشگر دو هسته‌ای) و درنتیجه مناسب نبودن کارایی در برخی نمونه‌ها دخیل هستند ازجمله انتقال کد و داده بین کلاینت و کارگرها و همچنین رقابت منابع بین فرآیندهای کارگر و فرآیندهای سیستم‌عامل. همچنین همان‌طور که در شکل ۵ مشخص است با افزایش اندازه مسئله سرعت این الگوریتم بالاتر رفته است. البته علت کارایی پایین برای اندازه کوچک مسئله را با دلالت به این مطلب که باز و بسته کردن ابزار موازی‌سازی خود عملی زمان‌بر است، می‌توان توجیه کرد؛ بنابراین استفاده از این الگوریتم برای مسائل کوچک پیشنهاد نمی‌شود.

ب) ارزیابی الگوریتم پیشنهادی با روش مقایسه تک ماشین

در این روش مقایسه برای مسئله موردبحث، الگوریتم‌های ژنتیک جزیره‌ای و سلولی با جعبه‌ابزار محاسبات موازی روی دو هسته پردازشگر به‌صورت موازی اجرا شدند و نتایج آن‌ها با الگوریتم ژنتیک استاندارد– که یک جمعیت دارد و تا رسیدن به‌شرط مشخص اعمال تکاملی را انجام می‌دهد– مقایسه شدند تا عملکرد سنجیده شود.

جدول ۴ میزان تسریع و کارایی الگوریتم‌های ژنتیک موازی برای اندازه‌های مختلف مسئله نمایش می‌دهد. بخش‌های الف و ب جدول ۴ به ترتیب عملکرد الگوریتم ژنتیک جزیره‌ای و سلولی را در سیستم توزیع‌شده‌ای با ۴ کامپیوتر نمایش می‌دهد.

جدول شماره (۴): مقایسه زمان اجرا، میزان تسریع و کارایی الگوریتم های ژنتیک موازی نسبت به الگوریتم ژنتیک استاندارد برای حل مسئله تخصیص ایستای وظایف در سیستم توزیع‌شده با ۴ کامپیوتر بر حسب ثانیه. (الف) زمان اجرا الگوریتم ژنتیک جزیره ای، میزان تسریع و کارایی آن نسبت به نسخه استاندارد. (ب) زمان اجرا الگوریتم ژنتیک سلولی، میزان تسریع و کارایی آن نسبت به نسخه استاندارد.

Efficiency	Speedup	T_{IGA}	T_{GA}	(M,N)
۰/۳۸	۰/۷۶	۰/۶۹	۰/۵۳	(۸،۴)
۰/۳۸	۰/۷۶	۰/۷۴	۰/۵۶	(۱۲،۴)
۱/۸۳	۳/۶۷	۲/۶۲	۹/۶۳	(۱۶،۴)
۱/۷۵	۳/۴۹	۱۶/۳۳	۵۷/۱۴	(۲۰،۴)

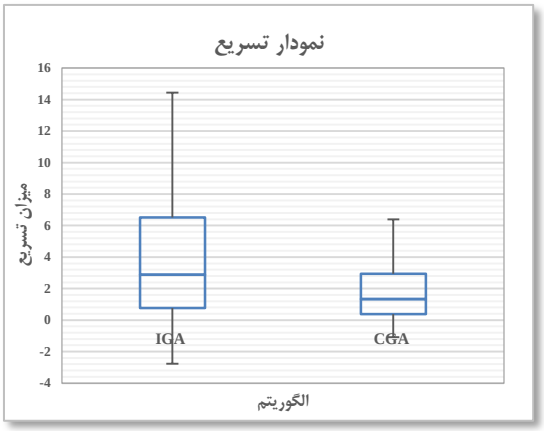
Efficiency	**Speedup**	**T_{CGA}**	**T_{GA}**	**(M,N)**
۰/۱۹	۰/۳۸	۱/۳۹	۰/۵۳	(۸،۴)
۰/۱۹	۰/۳۹	۱/۴۲	۰/۵۶	(۱۲،۴)
۰/۷۵	۱/۵	۶/۴	۹/۶۳	(۱۶،۴)
۰/۸۲	۱/۵۶	۳۴/۶۱	۵۷/۱۴	(۲۰،۴)

T_{GA}: زمان اجرای الگوریتم ژنتیک

T_{IGA}: زمان اجرای الگوریتم ژنتیک جزیره‌ای

T_{CGA}: زمان اجرا توسط الگوریتم ژنتیک سلولی

نتایج بدست آمده در بخش جدول ۴ نشان می‌دهد که هر دو الگوریتم ژنتیک جزیره‌ای در نوع مقایسه تک ماشین برای مسائل بزرگ می‌تواند به تسریع فراخطی[۳۱] دست یابد. آزمایش‌ها نشان داد الگوریتم ژنتیک استاندارد نسبت به الگوریتم‌های ژنتیک موازی به تعداد تکرار بیشتری برای رسیدن به جواب نیاز دارد و همین عامل باعث افزایش زمان اجرای آن می‌شود. دلایل نیاز الگوریتم ژنتیک استاندارد به تعداد تکرار می توان بخاطر علل ذیل بیان کرد: (۱) الگوریتم‌های ژنتیک موازی معمولاً سریع‌تر هستند و کمتر به پیدا کردن راه‌حل‌های زیربهینه تمایل دارند. (۲) الگوریتم ژنتیک استاندارد سرعتش کم است و به‌سرعت تنوع از دست می‌دهد. (۳) الگوریتم‌های ژنتیک موازی از نظر کارایی خیلی خوب عمل می‌کنند و در بسیاری از کاربردها خیلی سریع تکامل می‌یابند. شکل ۶ میزان تسریع دو الگوریتم موازی ژنتیک جزیره ای و ژنتیک سلولی را برای مسئله مورد بحث با روش مقایسه تک ماشین را نشان می دهد.



شکل شماره (۶): میزان تسریع الگوریتم های ژنتیک موازی برای مسئله تخصیص ایستای وظایف با روش مقایسه تک ماشین در یک سیستم توزیع شده با ۴ کامپیوتر.

شکل ۶ نشان می‌دهد که با استفاده از روش مقایسه تک ماشین، الگوریتم ژنتیک جزیره‌ای عملکرد بهتری نسبت به الگوریتم ژنتیک سلولی دارد و توانسته به تسریع فراخطی دست پیدا کند. الگوریتم ژنتیک جزیره‌ای به علت پیاده‌سازی مهاجرت، تنوع را در زیر جزایر آن افزایش می‌دهد و به پیدا کردن راه‌حل‌های بهتر کمک می‌کند؛ در مقابل در الگوریتم ژنتیک سلولی در هر نسل بایستی عملگر تقاطع روی تک‌تک سلول‌های آن اعمال شود و با توجه به زیاد بودن تعداد سلول، هر سلول مدت‌زمان زیادی را باید در انتظار بماند تا عملگرهای ژنتیک به آن اعمال شود.

ج) سایر مشاهدات

آزمایش‌های دیگری نیز انجام شد. برای مسئله مذکور الگوریتم ژنتیک جزیره‌ای و سلولی بدون سخت‌افزار موازی و به‌صورت ترتیبی روی یک هسته پردازنده اجرا و با الگوریتم ژنتیک استاندارد مقایسه شدند. نتایج تجربی نشان داد که الگوریتم ژنتیک استاندارد معمولاً به تعداد تکرار بیشتری (معمولاً دو یا سه برابر) نسبت به الگوریتم ژنتیک موازی برای رسیدن به جواب نیاز دارد و همین افزایش تعداد تکرارها، زمان اجرای آن را افزایش می‌دهد.

هر دو الگوریتم ژنتیک موازی که بدون موازی‌سازی سخت‌افزاری و به‌صورت ترتیبی اجرا شدند در تعداد تکرار کمتر و درنتیجه زمان کمتری به جواب رسیدند؛ اما خصوصاً الگوریتم ژنتیک جزیره‌ای بدلیل عملکرد بهتر درنتیجه، زمان بسیار کمتری به جواب می‌رسد. این عملکرد بهتر را می‌توان بعلت انجام عمل مهاجرت بین زیر جمعیت‌ها دانست زیرا مهاجرت به زیرجمعیت‌ها اجازه می‌دهد که مواد ژنتیکی را به اشتراک بگذارند و تنوع را در زیر جمعیت‌ها افزایش می‌دهد.

ه) نتیجه گیری و کارهای آینده

[31]. Super linear speedup

استفاده از الگوریتم‌های ژنتیک موازی برای مسئله تخصیص ایستای وظایف در سیستم‌های توزیع‌شده با شرط کاهش هزینه کل شامل هزینه اجرا و هزینه ارتباط در این پژوهش بحث شد. الگوریتم‌های ژنتیک موازی به‌طور موفقیت‌آمیز به این مسئله اعمال شدند و توانستند راه‌حل‌های بهینه و نزدیک بهینه برای مسئله مذکور پیدا کنند. برای بررسی عملکرد الگوریتم‌های موازی، ارزیابی با دو روش مقایسه رسمی و تک ماشین انجام شد. این الگوریتم‌ها در مسائل با مقیاس بالا عملکرد خوبی از خود نشان دادند و نشان داده شد که استفاده از آن‌ها در مسائل با مقیاس کوچک مقرون به‌صرفه نیست. به علاوه نتایج نشان داد استفاده از الگوریتم ژنتیک موازی حتی زمانی که الگوریتم روی یک هسته از یک پردازنده اجرا می‌شوند، نه فقط به الگوریتم‌های سریع‌تر، بلکه اغلب به عملکرد عددی بهتر نیز منجر می‌شود بااین‌وجود نکته جالب این است استفاده از جمعیت ساختاریافته در قالب جزایر یا در قالب در یک شبکه توری شکل مسئول چنین منافع عددی است.

بنظر می‌رسد اجرا برنامه روی یک کلاستر از کامپیوترها نتیجه بهتری داشته باشد و به تسریع فراخطی دست یابد و منابع فیزیکی به‌عنوان یک دلیل ممکن برای تحقق این امر در نظر گرفته می‌شود. هنگام اجرا روی یک کلاستر ممکن است منابع بیشتری برحسب حافظه یا کش در اختیار برنامه قرار بگیرد و هنگام انتقال کد از یک ماشین تک به یک کلاستر از کامپیوترها، الگوریتم – احتمالاً– از این منابع اضافی استفاده می‌کند. همچنین هر ماشین ممکن است فقط با یک بسته کوچک‌تر از داده سروکار داشته باشد و ممکن است داده‌های کوچک‌تر در کش جا شوند درحالی‌که در یک ماشین تک این‌گونه نیست و این می‌تواند تفاوت کارایی چشمگیری را ایجاد کند. اجرای روش پیشنهادی روی کلاستری از کامپیوترها و همچنین توسعه روش فوق برای حل دیگر ملاک‌های عملکرد یا حتی مسئله تخصیص چند هدفه به‌عنوان کارهای آینده توصیه می‌شود. پیشنهاد می‌شود این روش به‌منظور بالا بردن قابلیت اعتماد سیستم و هچنین توازن بار روی پردازنده‌ها استفاده شود.

٤- منابع

1. Aggarwal, A., Verma, R., & Singh, A. (2018a). An efficient approach for resource allocations using hybrid scheduling and optimization in distributed system. *International Journal of Education and Management Engineering, 8*(3), 33.
2. Aggarwal, A., Verma, R., & Singh, A. (2018b). An efficient approach for resource allocations using hybrid scheduling and optimization in distributed system. *Int. J. Educ. Manag. Eng.(IJEME), 8*(3), 33-42.
3. Akbari, M., & Rashidi, H. (2016). A multi-objectives scheduling algorithm based on cuckoo optimization for task allocation problem at compile time in heterogeneous systems. *Expert Systems with Applications, 60*, 234-248.
4. Alba, E. (2002). Parallel evolutionary algorithms can achieve super-linear performance. *Information Processing Letters, 82*(1), 7-13.
5. Alba, E., & Troya, J. M. (1999). A survey of parallel distributed genetic algorithms. *Complexity, 4*(4), 31-52.
6. Attiya, G., & Hamam, Y. (2003a). *Hybrid Algorithm for Mapping Parallel Applications in Distributed Systems.* Paper presented at the Fifth International Conference on PRAM, Poland.
7. Attiya, G., & Hamam, Y. (2003b). *Optimal allocation of tasks onto networked heterogeneous computers using minimax criterion.* Paper presented at the Proceedings of International Network Optimization Conference (INOC, 03).
8. Attiya, G., & Hamam, Y. (2006). Task allocation for maximizing reliability of distributed systems: A simulated annealing approach. *Journal of parallel and distributed computing, 66*(10), 1259-1266.
9. Augustine, D. P., & Raj, P. (2016). Performance Evaluation of Parallel Genetic Algorithm for Brain MRI Segmentation in Hadoop and Spark. *Indian Journal of Science and Technology, 9*(48).
10. Bharati, R., Shahakar, M., & Mahajan, R. (2014). Task Allocation Policy in Distributed Computing Using Refined Heuristics. *International Journal of Emerging Technology and Advanced Engineering, 4*(6).

11. Bharati, R. D., Jagtap, V. N., Gupta, O. C., & Landge, S. S. (2013). Task Allocation for Maximizing Reliability of Distributed Computing Systems using Dynamic Greedy Heuristic. *International Journal of Advanced Research in Computer and Communication Engineering, 2*(3), 1554-1557.
12. Braun, T. D., Siegel, H. J., Maciejewski, A. A., & Hong, Y. (2008). Static resource allocation for heterogeneous computing environments with tasks having dependencies, priorities, deadlines, and multiple versions. *Journal of parallel and distributed computing, 68*(11), 1504-1516.
13. Cai, P., Cai, Y., Chandrasekaran, I., & Zheng, J. (2016). Parallel genetic algorithm based automatic path planning for crane lifting in complex environments. *Automation in Construction, 62*, 133-147.
14. Chaubey, M., & Gupta, M. (2019). DESIGNING A TASK ALLOCATOR FRAMEWORK FOR DISTRIBUTED COMPUTING. *International Journal of Advanced Research in Computer Science, 10*(4).
15. Feng, Z.-K., Niu, W.-J., Zhou, J.-Z., Cheng, C.-T., Qin, H., & Jiang, Z.-Q. (2017). Parallel multi-objective genetic algorithm for short-term economic environmental hydrothermal scheduling. *energies, 10*(2), 163.
16. Govil, K. (2011). A smart algorithm for dynamic task allocation for distributed processing environment. *International Journal of Computer Applications, 28*(2), 13-19.
17. Govil, K., & Kumar, A. (2011). A modified and efficient algorithm for Static task assignment in Distributed Processing Environment. *International Journal of Computer Applications, 23*(8), 1-5.
18. Hamed, A. Y. (2012). Task allocation for minimizing cost of distributed computing systems using genetic algorithms. *International Journal of Advanced Research in Computer Science and Software Engineering, 2*(9).
19. Hu, Y., Li, J., & He, L. A reformed task scheduling algorithm for heterogeneous distributed systems with energy consumption constraints. *Neural Computing and Applications*, 1-13.
20. Hu, Y., Li, J., & He, L. (2019). A reformed task scheduling algorithm for heterogeneous distributed systems with energy consumption constraints. *Neural Computing and Applications*, 1-13.
21. Jiang, Y. (2015). A survey of task allocation and load balancing in distributed systems. *IEEE transactions on Parallel and Distributed Systems, 27*(2), 585-599.
22. Jiang, Y. (2016). A survey of task allocation and load balancing in distributed systems. *IEEE transactions on Parallel and Distributed Systems, 27*(2), 585-599.
23. Kacprzyk, J., & Pedrycz, W. (2015). *Springer handbook of computational intelligence*: Springer.
24. Kang, Q.-M., He, H., Song, H.-M., & Deng, R. (2010). Task allocation for maximizing reliability of distributed computing systems using honeybee mating optimization. *Journal of Systems and Software, 83*(11), 2165-2174.
25. Kang, Q., He, H., & Wei, J. (2013). An effective iterated greedy algorithm for reliability-oriented task allocation in distributed computing systems. *Journal of parallel and distributed computing, 73*(8), 1106-1115.
26. Khan, F. N., & Govil, K. (2013a). Cost Optimization Technique of Task Allocation in Heterogeneous Distributed Computing System. *International Journal of Advanced Networking and Applications, 5*(3), 1913.
27. Khan, F. N., & Govil, K. (2013b). Static Approach for Efficient Task Allocation in Distributed Environment. *International Journal of Computer Applications, 81*(15), 19-22.
28. Menghani, G. (2010). *A fast genetic algorithm based static heuristic for scheduling independent tasks on heterogeneous systems.* Paper presented at the Parallel Distributed and Grid Computing (PDGC), 2010 1st International Conference on.
29. Neelakantan, P., & Sreekanth, S. (2016). Task allocation in distributed systems. *Indian Journal of Science and Technology, 9*(31).
30. Sharma, G., & Martin, J. (2009). MATLAB®: a language for parallel computing. *International Journal of Parallel Programming, 37*(1), 3-36.
31. Shatz, S. M., Wang, J.-P., & Goto, M. (1992). Task allocation for maximizing reliability of distributed computer systems. *IEEE Transactions on Computers, 41*(9), 1156-1168.
32. Srinivasan, S., & Jha, N. K. (1999). Safety and reliability driven task allocation in distributed systems. *IEEE transactions on Parallel and Distributed Systems, 10*(3), 238-251.

33. Stone, H. S. (1977). Multiprocessor scheduling with the aid of network flow algorithms. *IEEE transactions on Software Engineering*(1), 85-93.
34. Sudholt, D. (2015). Parallel evolutionary algorithms *Springer Handbook of Computational Intelligence* (pp. 929-959): Springer.
35. Tyagi, R., & Gupta, S. K. (2018). A Survey on Scheduling Algorithms for Parallel and Distributed Systems *Silicon Photonics & High Performance Computing* (pp. 51-64): Springer.
36. Vidyarthi, D. P., & Tripathi, A. K. (2001). Maximizing reliability of distributed computing system with task allocation using simple genetic algorithm. *Journal of Systems Architecture, 47*(6), 549-554.
37. Wu, J. (2017). *Distributed system design*: CRC press.
38. Yadav, P., Singh, M., & Sharma, K. (2011). An optimal task allocation model for system cost analysis in heterogeneous distributed computing systems: A heuristic approach. *International Journal of Computer Applications, 28*(4), 30-37.
39. Yadav, P. K., Singh, M., & Sharma, K. (2011). Task Allocation Model for Reliability and Cost optimization in Distributed Computing System. *International Journal Of Modeling, Simulation, and Scientific Computing, 2*(02), 131-149.
40. Zhang, X.-Y., Zhang, J., Gong, Y.-J., Zhan, Z.-H., Chen, W.-N., & Li, Y. (2016). Kuhn–Munkres parallel genetic algorithm for the set cover problem and its application to large-scale wireless sensor networks. *IEEE Transactions on Evolutionary Computation, 20*(5), 695-710.

Static Task Allocation in Distributed Systems Using Parallel Genetic Algorithm

Monire Taheri Sarvetamin
Department of Computer Engineering, Islamic Azad University, Kerman, Iran
Email: mtaheri@iauk.ac.ir

Abstract
Over the past two decades, PC speeds have increased from a few instructions per second to several million instructions per second. The tremendous speed of today's networks as well as the increasing need for high-performance systems has made researchers interested in parallel and distributed computing. The rapid growth of distributed systems has led to a variety of problems. The most important problem that has been addressed by many researchers is the task allocation in such environments in order to obtain effective system efficiency. The task allocation problem is, except in a few specific cases, an NP-complete problem; so, heuristic methods are used to achieve suboptimal solutions in the desired time. Although different methods have been used in research, finding an effective and efficient method for this problem is still needed and desirable. This study used a parallel genetic algorithm to find the optimal solution for allocating a graph of tasks to the processors in a distributed system. The results showed that the proposed algorithm can provide optimal or near-optimal allocations for problems of different sizes. Also, the proposed method was able to solve problems of large and medium-size in a much faster time than traditional genetic algorithm with super linear speedup.
Keywords: Distributed System, Task Allocation, Parallel Genetic Algorithm, Island Genetic Algorithm, Cellular Genetic Algorithm.