To Appear, Proceedings of 7th Electrical Engineering Conference, May 17-19, 1999

پارلیدا: کتابخانه‌ای برای پردازش موازی در کاربردهای هندسه‌ی محاسباتی

محمّد قدسی مهدی شریف‌زاده

آزمایشگاه پردازش موازی، دانشکده مهندسی کامپیوتر
دانشگاه صنعتی شریف

چکیده

پارلیدا کتابخانه‌ای است که عملگرهای پایه‌ای برای پیاده‌سازی کاربردهای موازی هندسه‌ی محاسباتی را فراهم می‌آورد. این کتابخانه در طراحی و پیاده‌سازی این عملگرها، یک مدل موازی ناهمگن به نام اچ‌بی‌اس‌پی را که بر اساس مدل بی‌اس‌پی پیشنهاد کرده‌ایم، به‌کار می‌گیرد. دو کتابخانه‌ی لیدا و ام‌پی آی به‌ترتیب برای فراهم آوردن امکانات هندسی و موازی در پیاده‌سازی پارلیدا به‌کار گرفته شده‌اند. موازنه‌ی پویای بار در پردازنده‌های درگیر اجرای کاربرد موازی و توانایی تکرار شئ‌های زبان C++ در این پردازنده‌ها از ویژگی‌های دیگر سیستم است. پارلیدا به زبان C++ بر روی سیستم‌عامل Linux برای استفاده در کاربردهایی که به این زبان نوشته می‌شوند پیاده‌سازی شده است.

کلمات کلیدی: هندسه‌ی محاسباتی، پردازش موازی، موازنه‌ی بار، لیدا، ام‌پی آی، پارلیدا.

۱ مقدمه

بالا بردن کارایی از راه به‌کارگیری نیروی پردازشی بیشتر برای انجام محاسبات همیشه مورد توجه کارشناسان شاخه‌های گوناگون علوم بوده است. مسائل هندسه‌ی محاسباتی نیز به دلیل حجم بالای محاسبات و ورودی/خروجی کاندید خوبی برای به‌کارگیری پردازش موازی برای حل آن‌ها بوده‌اند تا جاییکه مطالعه‌ی آزمایش محک معماری‌های دارپا [1] در سال ۱۹۸۶ چهار مسئله‌ی هندسه‌ی محاسباتی را در سیاهه‌ی یازده مسئله‌ای که برای ارزیابی کارایی معماری‌های موازی در نظر گرفته بود جای داد [۴].

در این مقاله، هدف پیاده‌سازی یک کتابخانه‌ی نرم‌افزاری است که بتواند با فراهم آوردن تابع‌هایی عمومی برای برنامه‌نویسان الگوریتم‌های موازی هندسه‌ی محاسباتی آن‌ها را از درگیر شدن با جزئیات پیاده‌سازی موازی برهاند. در این راستا باید الگوریتم‌های موازی شناخته شده‌ی کارا مورد بررسی قرار گیرند و عملگرهای پایه‌ای لازم برای به‌کارگیری پردازش موازی در آن‌ها شناسایی شوند. این عملگرها می‌توانند در یک شکل عمومی برای الگوریتم‌های موازی هندسه‌ی محاسباتی تعریف شده و به عنوان تابع‌های واسط کاربری کتابخانه‌ی یادشده، در آن گنجانده شوند.

برای نمونه، در بسیاری از الگوریتم‌های موازی، در یک گام آغازین داده‌های ورودی مسئله که در مسائل هندسه‌ی محاسباتی شئ‌های هندسی هستند، باید بین پردازنده‌های درگیر حل مسئله پخش شوند. پیاده‌سازی یک تابع کتابخانه‌ای که بتواند این عمل را به کاراترین شکل انجام دهد از اهداف طراحی پارلیدا است. نمونه‌هایی از این‌گونه عملگرها در خلال بررسی تعدادی الگوریتم‌های موازی هندسه‌ی محاسباتی مانند یافتن

[1] DARPA Architecture Workshop Benchmark Study

پوسته‌ی کوژ نقاط در فضای دو و سه بعدی[۲]، مثلث‌بندی[۳] یا الگوریتم‌های کاربردی مانند الگوریتم‌های جی آی‌اس[۴] شناسایی و در گام تهیه‌ی کتابخانه‌ی پارلیدا پیاده‌سازی شده‌اند.

در این مقاله به بررسی مفاهیم به‌کاررفته در طراحی و پیاده‌سازی کتابخانه‌ی پارلیدا می‌پردازیم. مدل محاسباتی موازی که پارلیدا بر اساس آن بنا شده است یک مدل ناهمگن به نام اچ بی اِس پی[۵] است که ناهمگن بودن واحدهای پردازشی را در مدل بی‌اس‌پی به‌کار گرفته است و به تفصیل شرح داده می‌شود. از آنجا که پردازنده‌های به‌کاررفته در حل مسئله‌های موازی با پارلیدا ایستگاه‌های کاری یک شبکه‌ی یونیکس می‌باشند و می‌توانند دارای تفاوت‌هایی در سرعت باشند، روش‌هایی برای موازنه‌ی هوشمندانه‌ی بار پردازشی بین آن‌ها نیز در پارلیدا پیاده‌سازی شده است که به بررسی جزئیات آن می‌پردازیم. در پایان نمونه‌ای از برنامه‌نویسی با پارلیدا برای توسعه‌ی کاربردهای موازی هندسه‌ی محاسباتی را شرح می‌دهیم.

با بررسی الگوریتم‌های موازی هندسه‌ی محاسباتی، می‌توان دریافت که در بیشتر آن‌ها تعدادی تابع که رفتار یکسانی دارند و تنها در داده‌گونه‌های به‌کارگرفته شده با یکدیگر متفاوتند، در جهت فراهم آوردن یک محیط موازی برای کاربرد موردنظر مورد استفاده قرار می‌گیرند. بیشتر این تابع‌ها در بسیاری از الگوریتم‌های موازی دیگر نیز به‌کار گرفته می‌شوند و مخصوص الگوریتم‌های هندسه‌ی محاسباتی نیستند. در طراحی پارلیدا، بر آن شدیم که یک تعریف مجرد و مستقل از داده و الگوریتم برای هر یک از این عملگرها ارائه کنیم. اندیشه‌ی مطرح‌کردن چنین تعریفی، از کار پژوهشی آقای موربن[۶] که در خلال پروژه‌ی کارشناسی ارشد خود در دانشگاه کارلتون[۷] کانادا [۵] انجام داده است، به‌وجود آمد. وی در پیشنهادی که برای رابط برنامه‌نویسی یک کتابخانه‌ی موازی که آن را پی‌لیدا[۸][۶] نامیده است، برخی از عملگرهای پایه‌ای موازی برای الگوریتم‌های هندسه‌ی محاسباتی را برشمرده است.

پی‌لیدا پیاده‌سازی نشده است و به همین دلیل این که این کتابخانه تا چه‌حد می‌توانست در پیاده‌سازی الگوریتم‌های موازی تأثیرگذار باشد مشخص نشد. شکی نیست که اندیشه‌های گنجانده‌شده در پی‌لیدا در طراحی پارلیدا به شکل کارایی مورد استفاده قرار گرفته‌اند ولی طراحی و پیاده‌سازی پارلیدا از نخستین گام یعنی بررسی الگوریتم‌ها تا ساخت رابط برنامه‌نویسی به شکل شئ‌گرا در خلال این پروژه به شکل مستقل انجام شده است.

۲ جایگاه پارلیدا در توسعه‌ی کاربردها

پارلیدا با مطالعه و بررسی الگوریتم‌های موازی هندسه‌ی محاسباتی و بازیابی عملگرهای پایه‌ای مشترک آن‌ها طراحی شده است. در این کتابخانه تابع‌های پایه‌ای برای بخش‌بندی دامنه‌ی مسئله‌ی موردبررسی و همچنین امکاناتی برای موازنه‌ی بار بین پردازنده‌های درگیر گنجانده شده است. در طراحی و پیاده‌سازی این کتابخانه تا جای امکان سعی شده است که قابلیت حمل و استقلال آن از معماری سخت‌افزاری فراهم شود. کتابخانه‌ی ام‌پی آی [۱] به عنوان یک استاندارد تبادل پیام که پیاده‌سازی‌های آن بر روی تمامی معماری‌های موجود وجود دارد و قابلیت حمل یکی از پایه‌های طراحی آن را تشکیل می‌دهد، بستر تبادل پیام برای به‌کارگیری موازی نیروی پردازشی پردازنده‌های موجود در یک شبکه‌ی مبتنی بر تی سی پی / آی‌پی را فراهم می‌کند. تابع‌های این کتابخانه، امکان به‌وجود آوردن یک محیط برنامه‌نویسی موازی برای پیاده‌سازی پارلیدا را فراهم می‌آورند و در بسیاری از موارد تابع‌های پارلیدا، تابع‌های تغییرشکل‌یافته‌ی ام‌پی آی برای به‌کارگیری در کاربردهای هندسه‌ی محاسباتی هستند.

از سوی دیگر پارلیدا بر روی کتابخانه‌ی لیدا[۲][۳] بنا شده است. لیدا یک کتابخانه‌ی نرم‌افزاری رایگان است که ساختمان‌داده‌ها و الگوریتم‌های متعارف هندسه‌ی محاسباتی و ترکیباتی را به شکلی کارا پیاده‌سازی کرده است. تابع‌های پایه‌ای پارلیدا برای بخش‌بندی فضا نیز بر اساس داده‌گونه‌های لیدا پیاده‌سازی شده‌اند و می‌توانند برای بخش‌بندی دامنه‌های تشکیل‌شده از هر داده‌گونه‌ی لیدا به منظور توسعه‌ی یک برنامه‌ی موازی داده‌ای به‌کار آیند. برنامه‌نویسان کاربردهایی مانند جی آی‌اس که از الگوریتم‌های هندسه‌ی محاسباتی بهره می‌برند نیز می‌توانند برای پیاده‌سازی کاربردهای موازی از پارلیدا و کتابخانه‌هایی که امکانات خاص آن کاربرد را در اختیار قرار می‌دهند در کنار یکدیگر استفاده کنند. به بیان دیگر حتی می‌توان بر روی پارلیدا، کتابخانه‌های خاص‌منظوره‌ای برای توسعه‌ی کاربردهای موازی جی آی‌اس یا دینامیک شاره‌ها طراحی و پیاده‌سازی نمود. پارلیدا به عنوان یک واسط ارتباطی، برنامه‌نویس را از درگیری با مسائل تبادل داده‌ها بین پردازنده‌ها یا موازنه‌ی بار بین آن‌ها می‌رهاند و به تمرکز بیشتر وی روی بخش‌های خاص کاربرد کمک می‌کنند. ساختار لایه‌لایه‌ی

---

[۲] Convex Hull
[۳] Triangulation
[۴] GIS
[۵] HBSP: Heteregeneous Bulk Synchronous Parallel
[۶] Morin
[۷] Carleton University
[۸] PLEDA

شکل ۱: جایگاه پارلیدا در پیاده‌سازی کاربردهای موازی

نمایش داده شده در شکل ۱ در توسعه‌ی یک کاربرد موازی مرتبط با هندسه‌ی محاسباتی برای ارائه‌ی یک برنامه‌ی قابل‌حمل و پیمانه‌بندی شده مفید است. بهره‌گیری از پیمانه‌های جدا در این ساختار و پیاده‌سازی بهینه‌ی ارتباط بین هر دو لایه‌ی همسایه به بهتر شدن طرح کلی رهنمون می‌شود.

۳ مدل محاسباتی

توابع عملیاتی پارلیدا بر اساس یک مدل محاسباتی موازی ناهمگن که توسعه‌یافته‌ی مدل بی‌اس‌پی است و آن را اچ‌بی‌اس‌پی خواهیم نامید طراحی شده‌اند. اندیشه‌ی طراحی و استفاده‌ی چنین مدلی از [۵] برگرفته شده است که در این مرجع مدلی مشابه به نام اچ‌سی‌جی‌ام[۹] مورد استفاده قرار گرفته است. در اچ‌بی‌اس‌پی، پردازنده‌های موجود در ماشین موازی از نظر سرعت یکسان نیستند و این تفاوت در سرعت در مدل در نظر گرفته شده است. یک ماشین موازی همگام توده‌ای ناهمگن (اچ‌بی‌اس‌پی) از p پردازنده‌ی ناهمگن P_0، P_1، ... و P_{p-1} با سرعت‌های S_0، S_1، ... و S_{p-1} که اعداد صحیح می‌باشند تشکیل شده است. پارامتر $S = \sum_{i=0}^{p-1} S_i$ شناسای سرعت کلی ماشین موازیست.

هر پردازنده‌ی P_i کار معادل W واحد را در زمان W/S_i انجام می‌دهد و ارزش سرعت تمام پردازنده‌های دیگر ماشین موازی را می‌داند. S^{max} و S^{min} که سرعت سریع‌ترین و کندترین پردازنده‌ی ماشین هستند به شکل زیر تعریف می‌شوند:

$$S^{max} = max\{S_i : 0 \leq i \leq p-1\}$$

$$S^{min} = min\{S_i : 0 \leq i \leq p-1\}$$

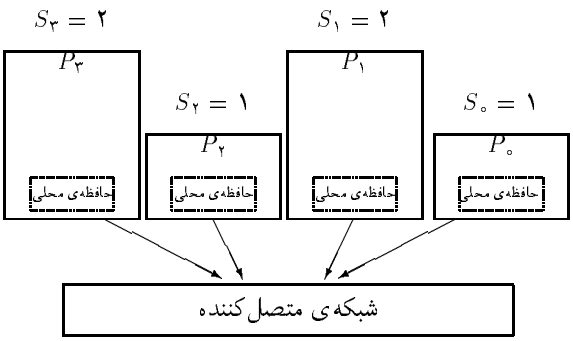
به همین ترتیب $P^{max} = P_{min\{i:S_i=S^{max}\}}$ و $P^{min} = P_{min\{i:S_i=S^{min}\}}$ سریعترین و کندترین پردازنده‌ی ماشین می‌باشند. نمونه‌ای از یک اچ‌بی‌اس‌پی با ۴ پردازنده را در شکل ۲ می‌بینید. در این نمونه، $P^{max} = P_1$، $S^{max} = 2$، $P^{min} = P_0$ و $S^{min} = 1$ می‌باشد. در حالت خاصی که $S_0 = S_1 = \ldots = S_{p-1}$ باشد، ماشین اچ‌بی‌اس‌پی تبدیل به یک ماشین بی‌اس‌پی می‌شود.

دو پارامتر g و l نیز که در مدل بی‌اس‌پی دخالت دارند برای پردازنده‌های با سرعت‌های نابرابر تفاوت دارند و باید به شکل میانگین روی پردازنده‌های متفاوت ماشین موازی تعریف شوند. هر پردازنده حافظه‌ی محلی خود را دارد که اندازه‌ی آن متناسب با سرعت آن است. با توجه به ناهمگونی پردازنده‌های ماشین، توزیع یکسان داده‌های مسئله بین آن‌ها منجر به نامتوازن شدن بار آن‌ها می‌گردد و برای بهره‌بردن از سرعت پردازنده‌های سریع‌تر باید به آن‌ها سهم بیشتری از داده‌ها نسبت داد. درباره‌ی چگونگی بخش‌بندی ورودی بین پردازنده‌های ناهمگن در بخش موازنه‌ی بار سخن خواهیم گفت.

۴ بخش‌بندی و تبادل داده‌ها

بسیاری از الگوریتم‌های موازی هندسه‌ی محاسباتی بر اساس استفاده از موازات داده‌ای بنا شده‌اند. نیاز به تبادل داده‌های

[۹] HCGM: Heteregeneous Coarse Grained Multicomputer



شکل ۲: یک‌اچ‌بی‌اس‌پی با ۴ پردازنده

به‌دست آمده از یک بخش‌بندی متوازن بین پردازنده‌های درگیر، الگوهای ارتباطی متنوعی را می‌طلبد. همیشه دو نکته در بخش‌بندی دامنه‌ی مسئله دارای اهمیت هستند: ارتباط بین داده‌های دو بخش همسایه و اندازه‌ی داده‌های هر بخش.

یک رابطه‌ی $\leq$ که بر اساس گونه‌ی دامنه تعریف می‌شود، بین داده‌های دو بخش همسایه وجود دارد که مرزهای بین دو بخش را تعیین می‌کنند. برای نمونه، در بخش‌بندی فضای نقاط دو بعدی به دو زیرفضای راست و چپ با یک خط عمودی که نقاط را به دو بخش مساوی تقسیم می‌کند، یک رابطه‌ی $\leq$ بین مشخصه‌ی x نقطه‌ها مورد ارزیابی قرار می‌گیرد. در تمامی گونه‌های بخش‌بندی که در فصل پیشین بررسی شدند چنین رابطه‌ای وجود دارد.

اندازه‌ی داده‌های هر بخش، کلید اصلی یک بخش‌بندی متوازن است. برخی بخش‌بندی‌ها یکسان بودن تعداد داده‌های هر بخش را با رابطه‌ی یاد شده در هم می‌آمیزند و به بخش‌هایی یکپارچه که با یکدیگر از دید تعداد داده‌ها برابرند ولی شکل یکسان ندارند می‌رسند.

در پارلیدا سعی شده است برای گونه‌های آشنای بخش‌بندی و تبادل بخش‌ها امکاناتی فراهم شود. این بخش‌بندی‌ها که از بررسی الگوریتم‌های موازی هندسه‌ی محاسباتی بدست آمده‌اند، در ادامه‌ی مقاله شرح داده می‌شوند.

۱.۴ نمونه‌گیری تصادفی

در الگوریتم‌های موازی تصادفی از نمونه‌گیری تصادفی به عنوان ابزاری کارا استفاده می‌شود. نمونه‌گیری تصادفی، انتخاب یک زیرمجموعه‌ی تصادفی با اندازه‌ی $O(r)$ از میان داده‌های یک مجموعه با اندازه‌ی n است. این زیرمجموعه پس از انتخاب به یکی از پردازنده‌ها فرستاده می‌شود که این پردازنده محاسباتی را روی آن انجام داده، پاسخ را بین سایر پردازنده‌ها پخش می‌کند. با توجه به اینکه P^{max} یعنی سریع‌ترین پردازنده‌ی ماشین اچ‌بی‌اس‌پی توانایی انجام سریع‌تر محاسبه‌ی یادشده را داراست، در مدل اچ‌بی‌اس‌پی بهتر است که نمونه‌ی تصادفی به این پردازنده فرستاده شود. در زیر دو گام این نمونه‌گیری را می‌بینیم:

الگوریتم ۱

۱) هر پردازنده‌ی P_i با احتمال $\frac{r}{n}$ روی هر قلم‌داده‌ی محلی خودش برای بودن یا نبودن آن در نمونه تصمیم‌گیری می‌کند.

۲) قلم‌داده‌های انتخاب‌شده در هر پردازنده به یکی از پردازنده‌ها (P^{max}) فرستاده می‌شوند.

۲.۴ افراز خطی

اگر S مجموعه‌ی داده‌های مسئله باشد و رابطه‌ی $\leq$ یک رابطه‌ی ترتیب جزئی روی S تعریف نماید، افراز خطی مجموعه‌ی S، بخش‌بندی آن به p زیرمجموعه‌ی جداگانه‌ی $S_۰$، $S_۱$، ... و $S_{p-۱}$ است به طوریکه برای تمام $x \in S_i$ و $y \in S_j$ که $i < j$ است، رابطه‌ی $x \leq y$ درست باشد.

افراز خطی در بیشتر الگوریتم‌های موازی به‌کار می‌رود. مرتب‌سازی داده‌ها یک حالت خاص افراز خطی مجموعه‌ی داده‌هاست که همه‌ی زیرمجموعه‌ها در حافظه‌ی یک پردازنده نگهداری می‌شوند. الگوریتم زیر برای مدل اچ‌بی‌اس‌پی از روی الگوریتم مدل اچ‌سی‌جی‌ام در اینجا آورده شده است.

الگوریتم ۲

۱) هر پردازنده با به‌کارگیری الگوریتم نمونه‌گیری تصادفی، یک نمونه با اندازه‌ی $O(r)$ (r ثابت است) را انتخاب می‌کند و به پردازنده‌ی P^{max} می‌فرستد.

۲) P^{max} نمونه‌های دریافتی را مرتب می‌کند. فرض کنیم این نمونه‌ها به شکل مرتب $K_۰$، $K_۱$،...، $K_{pr-۱}$ در لیست جای می‌گیرند.

۳) P^{max} از میان نمونه‌ها، $p+1$ جداکننده‌ی splitter_0،...، splitter_p به شکل زیر تعیین می‌کند:

$$\text{splitter}_i = \begin{cases} -\infty & \text{if } i = 0 \\ K\lceil \sum_{j=0}^{i} r_j \rceil & \text{if } 0 < i < p \\ \infty & \text{if } i = p \end{cases}$$

r_j با محاسبه‌ی سهم پردازنده‌ی P_j از داده‌ای با اندازه‌ی pr به‌دست می‌آید.

۴) P^{max} جداکننده‌ها را بین تمام پردازنده‌ها پخش می‌کند.

۵) در هر پردازنده‌ی P_i، هر داده‌ی محلی x به یک بسته‌ی B_{ij} فرستاده می‌شود اگر و فقط اگر $\text{splitter}_j \le x \le \text{splitter}_{j+1}$.

۶) هر پردازنده‌ی P_i داده‌های بسته‌ی B_{ij} را به P_j بفرستد.

۳.۴ جایگشت تصادفی

جایگشت تصادفی، راه‌کاری برای موازنه‌ی بار پردازنده‌های یک ماشین موازی است که در خلال آن داده‌های مسئله به شکل تصادفی به پردازنده‌های درگیر نسبت داده می‌شوند. اگر پردازنده‌های ماشین با یکدیگر تفاوت کارایی نداشته باشند، باید پراکندگی این جایگشت در تمام پردازنده‌ها یکنواخت باشد. اما مدل اچ‌بی‌اس‌پی، دخالت دادن سرعت پردازنده‌ها در این پراکندگی را می‌طلبد. روش کار به شکل زیر است:

الگوریتم ۳

- هر پردازنده‌ی P_i، هر قلم‌داده‌ی خود را به شکل تصادفی به یکی از p بسته‌ی فرضی B_{i0}،...، B_{ip-1} می‌فرستد. احتمال فرستادن هر قلم‌داده به بسته‌ی B_{ij} برابر $Prob_j$ است. $(\sum_{j=0}^{p-1} Prob_j = 1)$
- پردازنده‌ی P_i بسته‌ی B_{ij} را به P_j می‌فرستد.

احتمال $Prob_j$ با روش‌های هوشمندانه‌ی موازنه‌ی بار، با توجه به پیچیدگی زمانی گام محاسباتی بعدی که باید روی این جایگشت‌ها اجرا شود محاسبه می‌شود.

۴.۴ مرتب‌کننده‌ی سراسری

پس از بخش‌بندی دامنه‌ی مسئله، در بسیاری از الگوریتم‌ها نیاز به مرتب کردن داده‌های محلی وجود دارد. الگوریتم مرتب‌کننده‌ی موازی زیر که از ترکیب افراز خطی و مرتب‌سازی ترتیبی بوجود می‌آید و در [۷] به‌کار رفته است، در پارلیدا پیاده‌سازی شده است:

الگوریتم ۴

۱) با به‌کارگیری الگوریتم افراز خطی، داده‌های مسئله به p زیرمجموعه افراز می‌شوند.

۲) هر پردازنده، زیرمجموعه‌ی خود را به شکل ترتیبی و همزمان با سایر پردازنده‌ها مرتب می‌سازد.

۵.۴ پخش‌کننده و جمع‌کننده

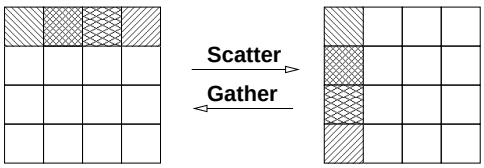
دو عملگر پخش‌کننده و جمع‌کننده که دوگان یکدیگر می‌باشند برای پخش بخش‌های به‌دست‌آمده از افراز دامنه و جمع‌آوری داده‌های محلی از پردازنده‌های دیگر توسط یک پردازنده‌ی ریشه به‌کار گرفته می‌شوند. در هر دوی این عملگرها یک پرداژه به‌عنوان ریشه معرفی می‌شود و ارسال یا دریافت داده‌ها تنها از طریق آن انجام می‌گردد. پخش و جمع‌آوری در شکل ۳ نمایش داده شده‌اند.

۵ رده‌های تکرارپذیر

داده‌گونه‌های هندسی لیدا در قالب شئ‌های زبان شئ‌گرای C++ پیاده‌سازی شده‌اند. بنابراین پارلیدا باید بتواند این شئ‌ها را بین پردازه‌های مختلف درگیر حل مسئله‌ی موازی جابجا کند. با توجه به اینکه تابع‌های فراهم شده در کتابخانه‌ی ام‌پی آی برای زبان C تهیه شده‌اند نمی‌توان از شئ‌های زبان C++ به عنوان پارامتر داده‌ی آن‌ها (مانند buf در تابع‌های بالا) استفاده کرد. به بیان دیگر این کتابخانه و همچنین کتابخانه‌ی پی‌وی‌ام نمی‌توانند در راستای بسته‌بندی، فرستادن و دریافت شئ‌های C++ به برنامه‌نویس کمکی بکنند و وی ناچار است برای این کار تمام عضوهای داده‌ای مخفی و عمومی این شئ ها را پس از بسته‌بندی با به‌کارگیری تابع‌های ارسال به پردازه‌ی دیگر بفرستد یا از یک پردازه‌ی دور دریافت کرده و در مقصد آن‌ها را بازسازی نماید. با این روند، بازسازی یک رونوشت از یک شئ که به تمامی برابر اصل باشد شاید تنها در سطح داده‌های دو شئ امکان‌پذیر باشد ولی تقلید رفتار شئ اصلی در پردازه‌ی دور بدین شکل غیرممکن است.

استفاده از توابع جابجایی داده‌گونه‌های پایه‌ای برای جابجایی شئ‌های یک زبان شئ‌گرا مانند C++ کارساز نیست، چراکه شئ‌ها علاوه بر داده دارای اطلاعات دیگری مانند متدها و جدولهای کنترلی مانند جدول وراثت[۱۰] هستند که رفتار پویای آن‌ها را تعیین می‌کند. وجود مفاهیمی مانند وراثت و یا

[۱۰] Inheritence



شکل ۳: عملگرهای پخش‌کننده و جمع‌کننده

چندشکلی‌بودن[۱۱]، جابجایی شئ‌ها را به کاری فراتر از ارسال داده‌های آن‌ها بین گره‌های ماشین موازی تبدیل می‌کند.

در مقایسه با کارهای انجام شده در این رابطه، پارلیدا یک راهکار عمومی را ارائه می‌دهد و رویه‌ای برای تبدیل هر رده به یک رده‌ی تکرارپذیر[۱۲] پیشنهاد می‌کند. برنامه نویس با مشتق‌کردن رده‌ی خود از یک رده‌ی تکرارپذیر و تعریف متدهای مجازی این رده بر اساس رویه‌ی ارائه شده، می‌تواند امکان استفاده از تابع‌های ام‌پی آی برای فرستادن شئ بوجود آمده از این رده و بازسازی آن در پردازه‌ی دیگر را فراهم سازد. از سوی دیگر امکان تنظیم تعداد و اندازه‌ی میانگیرهای لازم برای ارسال شئ تکرارپذیر، از بزرگ شدن اندازه‌ی پیام دربرگیرنده‌ی شئ می‌کاهد.

اندیشه‌ی کلی این است که رده‌هایی که آن‌ها را تکرارپذیر می‌نامیم پیاده‌سازی شوند که در بخش واسط[۱۳] خود داده‌هایی را که برای ایجاد یک رونوشت برابر اصل لازمند در دسترس قرار می‌دهند. این داده‌ها از طریق تعدادی متدهای مجازی در این رده‌ها تعریف شده‌اند. از سوی دیگر دو متد بسته‌بندی و بازسازی نیز که مستقل از داده‌ها از طریق این متدهای مجازی کار می‌کنند نیز پیاده‌سازی شده‌اند. حال برای این که امکان ایجاد رونوشت از یک شئ در سیستم ام‌پی آی وجود باشد، باید این شئ را از یک رده‌ی تکرارپذیر مشتق نمود [?].

۶ موازنه‌ی بار

در یک محیط موازی که بین پردازنده‌ها تفاوت سرعت وجود ندارد، بخش‌بندی دامنه‌ی مسئله به بخش‌هایی که اندازه‌ی داده‌های ورودی در آن‌ها برابر می‌باشد و توزیع این بخش‌ها بین پردازنده‌ها، باعث توازن بار پردازشی آن‌ها می‌شود. به همین دلیل در تمام الگوریتم‌های موازی بررسی‌شده برای طراحی پارلیدا سعی شده است که در گام بخش‌بندی دامنه، یک پارامتر در تمام بخش‌های بدست آمده یکسان باشد. این پارامتر گاه اندازه‌ی نقطه‌های موجود در بخش و گاه مساحت بخش است که بسته به گونه‌ی مسئله و دامنه‌ی آن تغییر می‌کند.

ولی در یک ماشین موازی که پردازنده‌ها ناهمگن باشند و سرعت آن‌ها متفاوت است، مساوی نگهداشت پارامتر یاد شده در بخش‌ها باعث بوجود آمدن اختلاف بار پردازشی پردازنده‌ها می‌شود. بدین ترتیب پردازنده‌های سریعتر زودتر از پردازنده‌های دیگر کار محاسبه روی بخش خود را به پایان می‌رسانند و به انتظار پایان محاسبه‌ی پردازنده‌های کندتر می‌مانند.

پردازنده‌ی P با سرعت S می‌تواند کار محاسبه‌ی W واحد کاری را در زمان $\frac{W}{S}$ انجام دهد. پارامتری که در بخش‌بندی‌های پارلیدا در نظر گرفته شده است، $Pt = \frac{W}{S}$ است. W که شناسای پیچیدگی زمانی یک گام محاسباتی است، تابعی از n یعنی اندازه‌ی ورودی مسئله می‌باشد.($W = W(n)$) اگر این پارامتر که برای هر پردازنده و بخشی که به آن تعلق می‌گیرد و محاسبه‌ی مورد نشر روی این بخش تعریف می‌شود بین تمام پردازنده‌های ماشین موازی یکسان باشد، انتظار می‌رود که همه‌ی پردازنده‌ها کار W را در یک زمان برابر به پایان برسانند.

بنابراین اگر n_i اندازه‌ی داده‌های بخشی از دامنه‌ی مسئله باشد که P_i محاسبات روی آن را انجام می‌دهد و $W_i = W(n_i)$ پیچیدگی زمانی انجام این محاسبات روی داده‌هایی با اندازه‌ی n_i باشد، برای اینکه تمام پردازنده‌ها محاسبات یاد شده روی داده‌های محلی خود را همزمان به پایان برسانند باید داشته باشیم:

$$\frac{W(n_0)}{S_0} = \frac{W(n_1)}{S_1} = \ldots = \frac{W(n_{p-1})}{S_{p-1}}$$

برای مثال برای اینکه زمان گام دوم یک عملگر مرتب‌سازی سراسری که از مرتب‌سازی‌های محلی تشکیل شده است در تمام پردازنده‌ها به یک اندازه طول بکشد، با توجه به اینکه پیچیدگی زمانی مرتب‌سازی $W(n) = nlogn$ است، باید داده‌ها به شکلی بین پردازنده‌ها توزیع شوند که رابطه‌ی زیر برقرار باشد:

$$\frac{n_0 logn_0}{S_0} = \frac{n_1 logn_1}{S_1} = \ldots = \frac{n_{p-1} logn_{p-1}}{S_{p-1}}$$

از طرفی می‌دانیم که $\sum_{i=0}^{p-1} n_i = n$ ارزشی معلوم دارد. بنابراین با دانستن ارزش S_i ها می‌توان این دستگاه را با p معادله

[۱۱] Polymorphism
[۱۲] Replicable
[۱۳] Interface

و p مجهول حل کرد و سهم هر پردازنده‌ی P_i (n_i) را از داده‌ها بدست آورد.

لم ۱. روی یک ماشین اچ‌بی‌اس‌پی با p پردازنده‌ی $P_0,\ldots,P_{p-1}$ با سرعت‌های $S_0,\ldots,S_{p-1}$، یک گام محاسباتی با درجه‌ی پیچیدگی $W = W(n)$، بر روی داده‌ای با اندازه‌ی n که بین پردازنده‌ها توزیع شده است و در تمام پردازنده‌ها به شکل همزمان آغاز شده است به شرطی در همه‌ی آن‌ها در یک زمان به پایان می‌رسد که داشته باشیم:

$$\frac{W(n_0)}{S_0} = \frac{W(n_1)}{S_1} = \ldots = \frac{W(n_{p-1})}{S_{p-1}}$$

برای پیدا کردن n_i، سهم پردازنده‌ی P_i، باید دستگاه p معادله و p مجهول زیر حل شود:

$$\begin{cases} \frac{W(n_0)}{S_0} = \frac{W(n_1)}{S_1} = \ldots = \frac{W(n_{p-1})}{S_{p-1}} \\ \sum_{i=0}^{p-1} n_i = n \end{cases}$$

n و S_i ها معلوم هستند.

چون حل این دستگاه هنگامی که $W(n)$ پیچیده باشد کار دشواریست، در پارلیدا یک راه‌حل تقریبی برای مسئله بدست می‌آید. با تعیین یک بازه برای پارامتر $\frac{W(n_i)}{S_i}$ و به‌کارگیری روش تقسیم و حل تا رسیدن به ارزشی که در دستگاه بالا صدق کند، سهم هر پردازنده مشخص می‌شود.

فرض می‌کنیم که سرعت پردازنده‌ها نسبت به شاخص آن‌ها افزایشی باشد یعنی داشته باشیم:

$$S_0 \le S_1 \le \ldots \le S_{p-1}$$

در نتیجه داریم:

$$n_0 \le n_1 \le \ldots \le n_{p-1}$$

بدیهی است که در مورد سهم کندترین و سریعترین پردازنده رابطه‌ی زیر درست است:

$$n_0 \le \frac{n}{p} \le n_{p-1}$$

چون $W(n)$، درجه‌ی پیچیدگی یک الگوریتم، نسبت به n یک تابع افزایشی است، رابطه‌ی زیر برقرار می‌باشد:

$$W(n_0) \le W(\frac{n}{p}) \le W(n_{p-1})$$

حال بازه‌ی یاد شده برای پارامتر $\frac{W(n_i)}{S_i}$ به شکل زیر بدست می‌آید:

$$\frac{W(n_0)}{S_0} \le \frac{W(\frac{n}{p})}{S_0}$$

$$\frac{W(n_{p-1})}{S_{p-1}} \ge \frac{W(\frac{n}{p})}{S_{p-1}}$$

یا به عبارت دیگر:

$$\frac{W(\frac{n}{p})}{S_{p-1}} \le \frac{W(n_0)}{S_0} = \ldots = \frac{W(n_{p-1})}{S_{p-1}} \le \frac{W(\frac{n}{p})}{S_0}$$

الگوریتم زیر با به‌کارگیری این دو حد برای پارامتر $\frac{W(n_i)}{S_i}$، سهم هر پردازنده در یک بخش‌بندی متوازن را پیدا می‌کند:

الگوریتم ۵

۱) پارامترهای Pt_1 و Pt_p به شکل زیر ارزش‌دهی می‌شوند:

$$Pt_1 = \frac{W(\frac{n}{p})}{S_{p-1}}$$

$$Pt_p = \frac{W(\frac{n}{p})}{S_0}$$

۲) $Pt = \frac{Pt_1 + Pt_p}{2}$ محاسبه می‌شود.

۳) با استفاده از رابطه‌ی $W(n_i) = S_i Pt$ برای $i = 0,\ldots,p-1$ ارزش n_iها بدست می‌آید.

۴) اگر $\sum_{i=0}^{p-1} n_i > n$ آنگاه $Pt_p = Pt$ و الگوریتم از گام ۲ دنبال می‌شود.

۵) اگر $\sum_{i=0}^{p-1} n_i < n$ آنگاه $Pt_1 = Pt$ و الگوریتم از گام ۲ دنبال می‌شود.

۶) در این گام رابطه‌ی $\sum_{i=0}^{p-1} n_i = n$ برقرار است و n_iها پاسخ مسئله هستند.

۷ واسط برنامه‌نویسی

هسته‌ی پارلیدا از یک رده به نام ParLeda تشکیل شده است که تمام توانایی‌های این کتابخانه را به عنوان متدهای خود در اختیار کاربر قرار می‌دهد. برنامه‌نویس در برنامه‌ی خود با تعریف یک شئ از گونه‌ی رده‌ی یادشده این توانایی‌ها را به‌کار می‌گیرد. به‌کارگیری متدهای پارلیدا تنها در محدوده‌ای از متن برنامه که بین فراخوانی دو متد Init و Finalize قرار گرفته است معتبر است. متد Init پارلیدا را آغازین‌دهی[۱۴] می‌کند و Finalize به کار سیستم پایان می‌بخشد. در بازه‌ی کاری پارلیدا باید ام‌پی آی آغازین‌دهی شده باشد. به بیان دیگر محدوده‌ی فعالیت پارلیدا زیرمجموعه‌ای از کد بین فراخوانی دو تابع MPI_Init و MPI_Finalize از کتابخانه‌ی ام‌پی آی می‌باشد.

برای موازنه‌ی بار، سرعت هر پردازنده به شکل پویا با استفاده از تعداد کارهای صف اجرای پردازنده و پارامترهایی

[۱۴] Initialization

مانند اندازه‌ی حافظه‌ی مجازی و میزان فضای مبادله[15] محاسبه و به‌کار گرفته می‌شود.

دو متد SetAlg و UnsetAlg که دوگان یکدیگر می‌باشند پیش و پس از انجام گام تبادل داده‌ها فراخوانی می‌شوند. این دو متد برای اعلام درجه‌ی پیچیدگی گام محاسباتی بعدی به سیستم به‌کار گرفته می‌شوند. برای نمونه به پیاده‌سازی یک مرتب‌سازی سراسری با استفاده از یک الگوریتم مرتب‌سازی ترتیبی با درجه‌ی پیچیدگی $O(nlogn)$ توجه کنید:

```
ParLeda pl(MPI_COMM_WORLD);

void GlobalSort(list<int> data) {
  pl.SetAlg(PL_NLOGN);
  pl.Partition(data, int_cmp);
  pl.UnsetAlg();
  data.sort();
}
```

درجه‌ی پیچیدگی الگوریتم با استفاده از ثابت‌هایی که در پارلیدا گنجانده شده است (مانند PL_NLOGN در نمونه‌ی بالا) به سیستم اعلام می‌شود.

۸ نتیجه‌گیری

در این مقاله به بررسی چگونگی طراحی و پیاده‌سازی کتابخانه‌ای نرم‌افزاری برای استفاده در کاربردهای موازی هندسه‌ی محاسباتی پرداختیم. با بررسی الگوریتم‌های موازی هندسه‌ی محاسباتی، عملگرهای مشترک شناسایی شدند. این عملگرها در تبادل داده‌های هندسی و توزیع متوازن آن‌ها بین پردازنده‌های درگیر به‌کار می‌آیند. الگوهای ارتباطی بین پردازنده‌ها به شکلی مجرد برای داده‌گونه‌های هندسی مختلف در پارلیدا پیاده‌سازی شده‌اند. یک رده‌ی سراسری که این الگوها را در متدهای خود جای داده است توسط برنامه‌نویس مورد استفاده قرار می‌گیرد. توانایی ایجاد توازن در بار پردازنده‌ها توسط این رده از ویژگی‌های برجسته‌ی سیستم است. این توازن به شکل پویا به هنگام بخش‌بندی دامنه‌ی ورودی مسئله بین پردازنده‌های ماشین موازی برپا می‌شود. ناهمگونی پردازنده‌ها ما را ناچار به طرح مدلی به نام اچ‌بی‌اس‌پی برای محاسبات موازی پارلیدا کرده است.

[15] Swap

مراجع

[1] MPI Forum, "MPI: A message passing interface", *Proc. of Supercomputing '93*, 878–883, November 1993.

[2] K. Mehlhorn and S. Naher, "LEDA: A Platform for Combinatorial and Geometric Computing", 1994.

[3] LEDA Home Page, http://www.mpi-sb.mpg.de/LEDA.html.

[4] A. Y. H. Zomaya (Edt.), "Parallel and Distributed Computing Handbook", McGraw-Hill, 1996.

[5] P. R. Morin, "Two Topics in Applied Algorithmics",
M.S. Thesis, School of CS., Carleton University, CA, 1998, Available through URL http://www.scs.carleton.ca/morin.

[6] P. R. Morin, "PLEDA User's Manual (v0.0)", Personal Communication, 12 Dec 1997.

[7] G. Hristescu, "Parallel Triangulation of a Set of Point for Coarse Grained Multicomputers", Departement of Computer Science, Rutgers University, October 1994.

[۸] مهدی شریف‌زاده و محمد قدسی، *شئ‌های تکرارپذیر در یک محیط توزیع‌شده*، گزارش فنی، آزمایشگاه پردازش موازی، دانشکده‌ی مهندسی کامپیوتر، دانشگاه صنعتی شریف، ۱۳۷۷.