بسمه تعالی

دانشگاه فردوسی مشهد

دانشکده مهندسی

گروه مهندسی کامپیوتر

پایان نامه کارشناسی ارشد

" موازنه بار در سطح منابع گرید با روش مدیریت منابع مبتنی بر عامل"

Grid Load Balancing in an Agent-Based Resource Management System

نویسنده:

محسن امینی صالحی

استاد راهنما:

دکتر حسین دلداری

استاد مشاور:

دکتر محمود نقیب زاده

دی ماه ۸٤

«صفحهٔ مربوط به نظرات اساتید محترم»

تقدیر و تشکر

ای که با نامت جهان آغاز شد　　دفتر ما هم به نامت باز شد

دفتری کز نام تو زینت گرفت　　کار آن از چرخ بالاتر گرفت

خداوندا، آنچه در این مختصر آمده را تو به من دادی و من هم به رسم امانت آنرا به تو تقدیم می کنم، باشد تا راهگشای کسانی باشد که در جهت فردایی بهتر برای بشریت گام برمی دارند.

خدایا تو را شکر که در راهی که می رفتم کسانی و نشانه هایی در راهم نهادی که مرا به سوی آنچه هدفم بود، هدایت کردند. در این میان لازم می دانم از اساتید با تجربه و پرتلاشم، جناب آقای دکتر دلداری و جناب آقای دکتر نقیب زاده که همواره راهنمای راهم بودند تشکر کنم. همچنین از همسر مهربانم که با نیروی عظیم عشق خود، سختی راهی که در بهار زندگی مشترک پیش رویمان بود را هموار نمود و آنچه گمان می رفت سد راهی باشد را به ابزاری امید بخش و یاری رسان بدل نمود، با تمام وجود متشکرم. تشکر ویژه ای از پدر و مادر عزیز و مهربانم دارم که دست یاریشان همواره همراهم بوده و هست. همچنین از کلیه کسانی که در طی مسیر زندگی بر آگاهی ام افزودند، به ویژه دوست خوبم جناب آقای مهندس عادل نجاران طوسی که همواره از ایده های ایشان سود برده ام تشکر و قدردانی می نمایم. این پایان نامه با پشتیبانی مالی مرکز تحقیقات مخابرات ایران به انجام رسیده است که در همین جا لازم می دانم کمال تشکر را از همکاری آنها داشته باشم.

خدایا در این سالها که گذشت، با چشمانی بازتر و تجربه ای بیشتر، یاری بی وقفه و بی منتت را در هجرت از هستها به سوی باید باشد های زندگی ام احساس کردم. اما به حکم انسان بودنم، به حکم نوری که از خود در من دمیدی، به حکم هجرتی که نصیبم کردی و نیاز ابدی که در من نهادی، هنوز نیازم به درگاه بیکرانت پایان نیافته و همواره از تو می خواهم که، چگونه زیستن را به من بیاموزی تا من خود چگونه مردن را بیاموزم.

الهی به امید تو.

چکیده

گرید محاسباتی یک محیط محاسباتی گسترده است که قدرت پردازشی عظیمی را برای پردازشهای توزیع شده فراهم می سازد. یکی از مسائل مهم در چنین محیطی مدیریت منابع می باشد. روشهای مبتنی بر عامل رویکرد مناسبی برای مدیریت منابع این محیطها می باشند. موازنه بار که یک قسمت مهم سیستم مدیریت منابع است تاثیر زیادی بر روی کارایی منابع موجود در گرید دارد. مدل مورچه ها یک روش اکتشافی خوب برای موازنه بار در سطح گرید می باشد ولی دارای نقائصی همچون حجم ارتباطات بالا می باشد. دسته ای دیگر از الگوریتمهای موازنه بار، به صورت دوره ای عمل می کنند و در سطحی محدود عملیات موازنه بار را انجام می دهند. ولی سرعت انتشار اضافه بار در آنها کند است. یک مشکل مشترک در بین روشهای موازنه بار، نبود تعریف دقیق و جامع برای مفهوم بارکاری است. با توجه به ویژگیهای ذکر شده و همچنین ویژگی منحصر به فرد مدل مدیریت منابع ARMS در گرید که تخمین میزان بار یک گره محاسباتی، قبل از اجرای پردازشهای آن را با کیفیت خوبی تضمین می کند، در این پایان نامه یک مدل چند سطحی برای موازنه بار در گرید و بر روی بستر ARMS ارائه شده است. در سطح اول این مدل، که سطح محلی نام دارد یک الگوریتم مبتنی بر الگوریتم ژنتیک، سعی در متوازن ساختن بار یک گره بین منابع داخلی آن می کند. در سطح دوم که سطح همسایگی نامیده می شود، از یک الگوریتم دوره ای استفاده می شود و از طریق آن گره های پربار، اضافه بار خود را در سطح ناحیه محدود اطراف پخش می کنند. در سطح سوم که سطح گرید نام دارد، اکوسیستمی از مورچه های هوشمند و خود مختار ارائه شده است. این لایه سعی در پخش بار در شعاع وسیعی در محیط گرید می نماید. این مورچه ها می توانند بنا به شرایط مورچه های جدیدی متولد کرده و یا بنا به شرایط محیط خودکشی کنند. یک ایده جدید به نام "موازنه بار در سطح مورچه ها" با الگوبرداری از زندگی مورچه ها نیز ارائه شده که باعث بهتر شدن کارایی کلی روش موازنه بار می شود. هر کدام از روشهای ارائه شده به طور جداگانه نیز یک روش موازنه بار محسوب می شوند، لیکن قرار گرفتن آنها در کنار هم در یک ساختار چند سطحی، باعث رفع نقائص آنها می شود و در نتیجه مدل چند سطحی حاصل که MLBM نام دارد، اکثر خواص یک سیستم موازنه بار را برآورده می سازد. در این مدل سعی شده از تعریف جدید و دقیقتری نیز برای مفهوم بار استفاده شود. در پایان نیز تحلیلهای تئوری و نتایج شبیه سازیها نشان می دهند که روش MLBM کارایی بهتری را نسبت به تک تک لایه ها و همچنین نسبت به سایر روشهای مشابه ارائه می دهد.

فهرست مطالب

فهرست اشکال

فصل ۱

مقـدمه

در پنجاه سال اخیر، سرعت کامپیوترها تقریباً یک میلیون برابر شده است. با این حال این کامپیوتر های ظاهراً پر سرعت امروزی در حل بسیاری از مسائل علمی روز دنیا، در یک زمان منطقی و قابل قبول هنوز نا توانند. برای مثال، در بعضی مسائل علم فیزیک، چند ابر رایانه برای تولید داده های اولیه لازم است. مسلماً تحلیل این داده ها احتیاج به قدرت محاسباتی بسیار بیشتری دارد. مثال دیگری از این دست، اطلاعات مربوط به کهکشانهاست که در هر لحظه توسط رصدخانه های پیشرفته در سرتاسر دنیا اطلاعات آنها دریافت می شود. مسلماً هیچ کامپیوتر روز دنیا نمی تواند این حجم فزاینده اطلاعات را در یک زمان منطقی پردازش کند.

یک راه حل برای مشکل نیاز به قدرت محاسباتی بالا، تحقیق بر روی محاسبات خوشه ای[۱] [۷٤] بود. در این سیستمها چندین کامپیوتر مجزا از هم به هم متصل شده و همزمان با هم شروع به حل یک مساله بزرگ می کردند. سیستم ASCI در ایالات متحده آمریکا نمونه ای از این گونه معماری است. هر گره این ابر کامپیوتر، یک چند پردازنده ای متقارن[۲] (SMP) با ۱٦ پردازنده است که توسط وسایل ارتباطی سریع به هم متصل شده اند. با اینکه راهکار کامپیوتر های خوشه ای پیشرفت قابل توجهی در به وجود آمدن قدرت محاسباتی بالا محسوب می شود، ولی باید توجه داشت که یک خوشه، ماشین مجزایی است که برای بر آوردن هدف خاصی در یک محدوده خاص وقف شده است. چنین سیستم هایی قابلیت استفاده در مقیاس بزرگ و بین سازمانی را نداشتند.

با رشد سریع راهکارهای ارتباطی، محاسبات بین شبکه ای[۳] [۳۰] در جهت ارائه توان محاسباتی بالاتر با توزیع شده گی بیشتر به وجود آمد. این روش سعی در استفاده از توان محاسباتی میلیونها کامپیوتر شخصی بیکار در سراسر اینترنت دارد. تصور می شود که این امکان وجود دارد که با این روش بسیاری از مسائل محاسباتی سنگین دانشمندان از طریق بستر اینترنت حل شود. البته باید توجه داشت که سرویسهای امروزی اینترنت همچون پست الکترونیک و خدمات وب برای چنین کاربرد هایی بسیار ساده و ابتدایی هستند و در عمل برای اجرای برنامه ها نیاز به سرویسهای پیچیده تری همچون امکانات ارتباط داده ها، کامپیوتر ها و سایر منابع است. در این صورت دانشمندان می توانند به وسیله اینترنت آزمایشگاههای مجازی[٤] [۲۹] داشته باشند و از طریق این آزمایشگاه ها می توانند برنامه های علمی خود را حتی بر روی منابع راه دور اجرا کنند. آنچه از آن تحت عنوان محاسبات گریدی یاد می شود در پی به انجام رساندن چنین هدفی است.

۱-۱-گرید محاسباتی

تمدن امروزی بشر مدیون زیرساختهایی است که به ویژه در طی قرن بیستم به وجود آمدند. جاده های ارتباطی، راه آهن، شبکه برق، سیستم تلفن و اینترنت نمونه هایی از این زیرساختها هستند. وقتی شما کلیدی را فشار می دهید و چراغی روشن می شود شما از ان استفاده می کنید بدون آنکه بدانید این انرژی از کجا می آید. اینترنت که از آن تحت عنوان بزرگ راه ارتباطات یاد می کنند، آخرین زیرساخت از این دست است که تا کنون به وجود آمده است. در اینترنت با دادن یک نام دامنه می توانید به حجم زیادی از اطلاعات دسترسی داشته باشید، بدون اینکه بدانید این اطلاعات از کجا آمده است.

---

[۱] Cluster Computing
[۲] Symmetric Multi Processor
[۳] Internet Computing
[٤] Virtual Lab

تا کنون اصطلاحاتی همچون "کامپیوتر شبکه"، "کامپیوتر گسترده جهانی" و "شبکه اطلاعات " [۲۸] را بارها شنیده ایم. در واقع دانشمندانی که بر روی مسائلی همچون ارائه قدرت محاسباتی بالا کار می کنند، با بیان این اصطلاحات مقدماتی در پی ارائه مفاهیم و راهکارهای جدید تری هستند که به وسیلهٔ آنها از طریق شبکه های عظیم کامپیوتری امروزی، نه تنها اطلاعات مورد نیاز انسانها را برآورده می شود، بلکه از طریق همان شبکه ها قدرت محاسباتی عظیمی نیز در اختیار کاربران خواهد بود. گرید محاسباتی، زیر ساختی نرم افزاری و سخت افزاری است که سعی در ارائه دسترسی سازگار، ارزان و گسترده به توان محاسباتی بالاست [۹۲]. گرید حاوی قراردادها[۱]، سرویسها و نرم افزار های لازم برای ایجاد دسترسی اشتراکی به منابع بهصورت انعطاف پذیر، کنترل شده و مقیاس پذیر است. اجزاء اصلی در معماری گرید عبارتند از[٥٤]:

- زیرساخت گرید[۲]- متشکل از منابع سراسری گرید که به لحاظ جغرافیایی توزیع شده اند و قابل دسترسی از طریق اینترنت هستند. این منابع است شامل کامپیوترها (مثل کامپیوتر های شخصی، ایستگاه های کاری و سیستم عاملهای در حال اجرا بر روی آنها مثل ویندوز و یونیکس)، ابر کامپیوتر ها، خوشه ها (که دارای سیستم عامل خاص و یا سیستمهای مدیریت منابعند)، بانکهای اطلاعاتی و سایر وسایل آزمایشگاهی و حتی صنعتی مورد نیاز آزمایشگاههای مجازی است.
- سرویسهای گرید[۳]- سرویسهای اصلی[٤] گرید مثل اطلاعات، ارتباطات، نام گذاری[٥]، مدیریت منابع، تحلیل کارایی، امنیت، هویت سنجی، حساب رسی و سایر موارد مرتبط را ارائه می دهد.
- ابزارهای گرید[٦]- سرویسهای سطح بالایی را ارائه می دهد که به برنامه نویسان امکان توسعه برنامه های تحت گرید را می دهد. این سرویسها شامل زبانها، واسطه های برنامه نویسی[۷]، ابزارهای توسعه[۸]، ابزارهای عیب یابی[۹] و سایر ابزارهای مربوط می باشد.
- برنامه های کاربردی گرید[۱۰]- این برنامه ها توسط ابزارهای گرید به وجود می آیند. برنامه های کاربردی تحت گرید انواع مختلفی همچون برنامه های ابر محاسباتی توزیع شده در حد وسیع[۱۱]، محاسبات با توان عملیاتی بالا[۱۲]، محاسبات با حجم داده های بالا[۱۳] و محاسبات نیازی[۱٤] را شامل می شود.

تحقیقات بر روی گریدهای محاسباتی در سه فاز زیر انجام شده اند:

- فاز اکتشاف[۱]- مربوط به سالهای قبل از سال ۱۹۹۸ برمی گردد، تلاشهای اولیه که امروزه به عنوان پروژه های کلاسیک در زمینهٔ گرید از آنها یاد می شود، انگیزه های اولیه متفاوتی داشته اندGUSTO[۲]، نمونه اولیه ای

---

[۱] Protocol
[۲] Grid Fabric
[۳] Grid Services
[٤] Core
[٥] Naming
[٦] Grid Tools
[۷] API: Application Programming Interface
[۸] SDK: Service Development Kit
[۹] Debugger
[۱۰] Grid Application
[۱۱] Wide-area distributed supercomputing
[۱۲] High-Throughput computing
[۱۳] Data-intensive computing
[۱٤] On-demand computing

گریدهای محاسباتی بعدی بود. همچنین چاپ کتاب "The GRID- Blueprint for New Computing Infrastructure" [۹۲]، نشان از مراحل آغازین به وجود آمدن مفهوم گرید دارد.

- فاز گسترش- بازه سالهای ۱۹۹۸-۲۰۰۱ را در بر می گیرد. در طی این دوره مفهوم گرید با سرعت زیادی گسترش یافت. محققان از منظر های متفاوت گرید را مورد نقد و بررسی قرار دادند. پروژه های تحقیقاتی بسیاری برروی قسمتهای مختلف مطرح در گرید شروع به کار کردند. در همین زمینه درماه مارس سال ۲۰۰۱، ۳٦۰ محقق از آمریکا، اروپا و ژاپن در اولین انجمن سراسری گرید (GGF[۱]) در آمستردام شرکت کردند. در ماه می همان سال ۲۰۰ نفر از محققان از سراسر جهان در اولین گردهمایی بین المللی IEEE/ACM در مورد محاسبات خوشه ای و گرید (CCGrid۲۰۰۱) در استرالیا حضور یافتند.
- فاز رشد انفجاری[۳]- این مرحله از سال۲۰۰۱ به بعد را شامل می شود. با ورود به هزارهٔ جدید، محاسبات گریدی تبدیل به یک زمینه باز برای تحقیقات در علوم کامپیوتر گشته است. بعضی از شرکتها نیز از این تحقیقات حمایت می کنند. دولتها نیز شروع به برنامه ریزی برای ایجاد گرید بومی کرده اند. برای مثال، اتحادیه اروپا ۹/۸ میلیون یورو در سه سال اخیر بر روی پروژه DataGrid [٥] سرمایه گذاری کرده اند. بخش صنعت انگلیس مبلغ زیادی را برای فعالیتهای مربوط به کاربرد گرید در علوم تخصیص داده است.

واضح است که زیرساخت نرم افزاری گرید پیچیده گیهای بسیار زیادی خواهد داشت. مهمترین قسمتهای گرید، سرویسهای آن هستند، که به عنوان میان افزاری بین منابع گرید و برنامه های کاربردی تحت گرید قرار می گیرند. در حال حاضر سیستمهای نرم افزاری زیادی بطور جداگانه و با انگیزه ها، روشها و ابزارهای متفاوت در حال گسترشند. برای یکپارچه کردن فعالیتهای فعلی و عملیاتی ساختن گرید، روشهای و تکنولوژیهای نرم افزاری پیشرفته باید برای توسعه گرید بکار روند.

۱-۲- عاملهای نرم افزاری

امروزه عاملهای نرم افزاری تبدیل به یک روش پیشرفته برای توسعهٔ نرم افزارها گشته اند. به وجود آمدن کاربردها و رویکردهای متفاوت در زمینه های مختلفی [٤٤]، همچون واسطهای کاربر هوشمند [۲٤]، تجارت الکترونیک [۲٦]، کتابخانه های دیجیتال [۹]، مدیریت شبکه [۷٥] و سایر موارد.

عاملها، سیستمهای کامپیوتری هستند که توانایی عملکرد قابل انعطاف، خود مختار در محیطهای غیر قابل پیش بینی را دارند. خود مختاری مهمترین این ویژگیهاست، که عاملها را از سایر روشهای نرم افزاری جدا می کند. متاسفانه همانطور که در [٦٤] اشاره شده است، خود مختاری مفهوم پیچیده ای است و نمی توان آن را به شکل مدون تعریف کرد، ولی منظور ما از آن، این است که سیستم بتواند بدون دخالت مستقیم فرد خارجی (یا عاملهای دیگر) کار خود را انجام دهد.

دو روش اساسی برای اعمال خود مختاری وجود دارد:

هوشمندی و رفتار اجتماعی. هوشمندی به معنای آن است که با اعمال روشهای هوش مصنوعی مثل شخصیت، احساس، یادگیری، استنتاج و غیره به یک عامل، آنرا خود مختار کرد. رفتار اجتماعی به این معنی است که یک عامل خود مختاری

---

[۱] Exploration
[۲] Globus Ubiquitous Supercomputing Test bed
[۳] Exploding

خود را با برقراری ارتباطاتی با سایر عاملهای موجود در محیط چند عامله به دست آورد. این ارتباطات ممکن است از طریق زبان ارتباط عاملها[۱] (ACL)، مذاکرات هماهنگی، مکانیزم فروش، تحرک و سایر روشها بدست آید.
عاملهای نرم افزاری در توسعه سیستمهای باز (سیستمهایی که دامنه تغییرات اجزا در آنها زیاد است)، سیستمهای پیچیده نرم افزاری و سیستمهای محاسباتی همه جایی[۲] کاربرد دارند. بنابراین شرایط می توان نتیجه گرفت که عاملها انتخاب مناسبی برای مدیریت منابع در محیط گرید هستند.

۱-۳- محیط مدیریت منابع ARMS

همانطور که در قسمت قبل بیان شد، عاملها ابزارهای مناسبی برای مدیریت منابع گرید هستند. تا کنون چند سیستم مدیریت منابع مبتنی بر عاملها به وجود آمده اند ولی جدی ترین کار انجام شده در این زمینه سیستم ARMS [۳۳] است که در سال ۲۰۰۱ در دانشگاه Warwick اجرا شده است. این سیستم مدیریت منابع از سلسله مراتبی از عاملهای همگون [۳۵] که مجهز به ابزاری برای پیشگویی کارایی، به نام PACE [٤١] می باشند، استفاده شده است.
PACE ابزاری قوی در پیشگوئی اطلاعات کارایی منابع و برنامه ها است. همچنین PACE دارای ابزاری به نام موتور ارزیابی است که می تواند مشخص کند یک برنامه خاص بر روی یک منبع خاص با چه کارایی اجرا می شود. ابزار PACE تا کنون در سیستمهای بسیاری مورد استفاده قرار گرفته و صحت و دقت کارایی آن اثبات شده است.
در این سیستم برای هر منبع یا کاربر یک عامل در نظر گرفته می شود. هر عامل می تواند در نقش ارائه دهنده منبع یا متقاضی منبع و یا حتی میانجی[۳] ظاهر شود. هر عامل اطلاعات منابعی را که مدیریت می کند و همچنین اطلاعات مربوط به منابع همسایه های خود را دارا می باشد. عامل این اطلاعات را در قالب جدولهایی نام ACT[٤] نگهداری می کند. این عاملها دارای تحرک نیستند بلکه در یک ساختار سلسله مراتبی نسبت به هم قرار می گیرند. تنها عاملی که ساختار آن با ساختار سایر عاملها متفاوت است، در ریشه این درخت قرار دارد و به آن PMA[۵] گفته می شود. وظیفه PMA بهینه سازی رفتار عاملها در جهت افزایش بهره وری سیستم مدیریت منابع است. به عبارت دیگر PMA پس خور محیط و اطلاعات آماری را از عاملها دریافت کرده و تصمیم گیریهایش را بر اساس آنها انجام می دهد.
هر گره ARMS همچنین دارای زمانبندی محلی به نام COSY [۳۹] است. این زمانبند قدرت زمانبندی فرآیندهای دسته ای و بلادرنگ را دارا می باشد.
روش کشف و ارائه منبع در ARMS به طور خلاصه به اینصورت است که با ورود یک تقاضا به یک عامل، ابتدا عامل به جداول ACT خود مراجعه می کند، در صورتیکه منبع مورد نظر توسط خود عامل قابل ارائه نبود و یا منبع مورد نظر با میزان کارایی مورد نیاز برنامه هماهنگ نبود (این اعتبار سنجی با استفاده از موتور ارزیابی PACE انجام می شود)، سپس عامل به جدول اطلاعات همسایه های خود مراجعه می کند تا شاید منبع مورد نیاز در آنها موجود باشد. در صورتیکه منبع مورد نظر در یکی از همسایه ها وجود داشت، عامل تقاضا را به آن عامل ارجاع می دهد. اگر با اجرای این روال تقاضا به ریشه درخت

---
[۱] Agent Communication Language
[۲] Ubiquitous
[۳] Matchmaker
[٤] Agent Capability Table
[۵] Performance Monitor and Advisor

رسید و هیچ منبع مناسبی را پیدا نکرد، در اینصورت تقاضا با شکست مواجه شده است. به این ترتیب ادعا شده که مدل ARMS تا حد زیادی مساله قیاس پذیری و تطبیق پذیری در گرید را که دو مساله مهم و اساسی به شمار می آیند، برطرف ساخته است.

نکته ضعفی که در سیستم ARMS وجود دارد این است که گرچه منابع مورد نیاز تقاضا ها در اکثر موارد با موفقیت توسط عاملها کشف می شوند ولی در اکثر اوقات از منابع موجود به شکل متوازنی استفاده نمی شود. این امر به دلیل عدم بکارگیری روالهای مخصوص موازنه بار در عاملها یا بین آنهاست.

۱-٤- موازنه بار در گرید

در یک تعریف کوتاه می توان گرید را یک سیستم توزیع شده بزرگ یا در واقع بدون مرز در نظر گرفت. یکی از مسائل مهم در سیستمهای توزیع شده زمانبندی کارها بر روی منابع موجود است. یک زمانبند خوب بایستی عملیات زمانبندی را به نحوی انجام دهد که دو هدف اصلی افزایش بهره وری منابع و کم شدن زمان انتظار تقاضاها برآورده شوند. به خاطر سنگین بودن و زمانبر بودن این وظیفه و همچنین اهمیت سرعت در عملکرد زمانبند، معمولاً در سیستمهای مدیریت منابع سیستمهای توزیع شده از روالهای مجزایی برای انجام آنها استفاده می شود. به این روالها، روالهای موازنه بار می گویند.

در یک تعریف کلی، هدف الگوریتمهای موازنه بار، توزیع یکسان بار بر روی منابع و حداکثر کردن کارایی منابع و در ضمن کم کردن زمان اجرای کلی کارهاست [٤]. یعنی اینکه اختلاف بین پربارترین و کم بارترین منابع باید حداقل باشد. مسئله موازنه بار برای محیط گرید که در آن یکی از مهمترین مسائل، توزیع عادلانه بار بر روی منابع می باشد نیز یک ضرورت اساسی محسوب می شود.

ویژگیهای یک راهکار موازنه بار مطلوب عبارتند از: قیاس پذیری[۱]،تطبیق پذیری[۲]، پایداری[۳]، شفافیت از دید برنامه کاربر[٤]، دارای قابلیت تحمل در برابر خطا[٥] و حداقل بودن هزینه سربار تحمیلی به سیستم [١٤].

به طور کلی روشهای موازنه بار به دسته های مرکزی و غیر مرکزی، ایستا یا پویا، دوره ای یا غیر دوره ای و نیز دارای حد آستانه و فاقد حد آستانه[٦] تقسیم بندی می شوند[۱۷، ۹۰، ۲۳].

به طور کلی فرآیند موازنه بار در الگوریتمهای پویا دارای روالهای اندازه گیری بار، تبادل اطلاعات، راه اندازی و انجام عملیات نهایی موازنه بار است. با توجه به ضرورت موازنه بار در محیط گرید و ضعف مدل مدیریت منابع ARMS در ارائه یک روش موازنه بار مناسب و کارا، در این پایان نامه سعی شده است روشی کارا برای موازنه بار ارائه شود.

---

[۱] Scalability
[۲] Adaptability
[۳] Stability
[٤] Application Transparency
[٥] Fault Tolerant
[٦] With Threshold & Without Threshold

۱-۵- تحقیق انجام شده در این پایان نامه

با توجه به آنچه در مورد سیستم مدیریت منابع ARMS و نقاط قوت و ضعف آن (به ویژه در زمینهٔ موازنه بار گره ها) گفته شد، در این پایان نامه سعی شده روش جدیدی برای موازنه بار در این سیستم مدیریتی ارائه شود. مدل موازنه بار ارائه شده که MLBM نام دارد یک مدل سه سطحی است که هر یک از سطوح آن به نوعی عملیات موازنه بار را انجام می دهند. باید توجه کرد که هر یک از این سطوح به طور مجزا نیز یک روش موازنه بار محسوب می شوند و حتی در فصل ٤ برای نشان دادن این مساله آزمایشهای جداگانه ای بر روی هر یک از آنها انجام شده است. دلیل استفاده از یک قالب سه سطحی، افزایش کارایی هر یک از روشها و همچنین افزایش سرعت همگرایی نهایی سیستم است. یک دلیل دیگر این است که ویژگیهای بیان شده برای موازنه بار در قسمت قبل (که بطور کاملتر در فصل ۳ نیز بیان شده اند) اغلب با هم در تضادند. یعنی برآوردن یک ویژگی باعث صدمه خوردن ویژگی (های) دیگر می شود، یا عملکرد بعضی الگوریتمها به گونه ای است که فقط یک یا چند مورد از آن ویژگیها را برآورده می کنند. با ترکیب سه سطح مختلف وظیفه کلی هر سطح به اینصورت است که در پایینترین سطح ابتدا بار روی پردازند ه های یک گره به نحو بهینه گسترده می شوند، تا کارایی هر گره تا حد امکان افزایش یابد. سطح دوم سطح همسایگی نام دارد و بار گره های پربار را در سطح یک ناحیه پخش می کند. و عدم توازن گرهی را کاهش داده، اضافه بار را در سطح یک ناحیه می گستراند. در سطح سوم الگوریتم مبتنی بر مورچه ها قرار دارد که به صورت سرسری عمل کرده و عدم توازنهای ناحیه ای را از بین می برد.

در زیر فعالیتهای انجام گرفته در راستای ارائه مدل MLBM به صورت کلی آورده شده است:

- ارائه مدل موازنه بار مبتنی بر گروه مورچه های هوشمند. در بالاترین سطح مدل موازنه بار ارئه شده از روش بهینه سازی مبتنی بر گروه مورچه ها استفاده کرده ایم. برای این منظور اکوسیستمی از مورچه های هوشمند ارائه داده ایم. منظور از اکوسیستم این است که مورچه ها در این سیستم خود به خود به وجود آمده و خود به خود (بدون دخالت کاربر) از بین می روند. یعنی تعداد مورچه ها متغیر بوده و بنا به شرایط سیستم تطبیق می شود. در ضمن تعداد گامهای مورچه ها هم در دوره های مختلف حیاتشان متفاوت بوده و بنا به وضعیت توازن کلی بار در سیستم تعیین می شود. مورچه ها دارای حافظه و دانش نسبت به محیط اطراف خود می باشند و بر اساس این دانش و با توجه به شرایط محیطی در هر گره تصمیم گیری متناسب با آن گره را انجام می دهند. همچنین حافظه مورچه ها باعث افزایش کارایی و سرعت همگرایی می شود. چرا که در هر دفعه انجام موازنه بار چندین گره پربار را با چندین گره کم بار موازنه می کنند. مورچه در شرایطی که احساس کنند وجود آنها سودمندی برای محیط ندارد، دست به خودکشی می زنند و این آخرین حلقه اکوسیستم ذکر شده است. عیب اصلی این الگوریتم حجم زیاد ارتباطات در آن و نا عادلانگی است.
- در سطح دوم از یک روش موازنه بار در سطح زمانبند استفاده شده است. این الگوریتم که در این پایان نامه آنرا "الگوریتم سطح همسایگی " نام گذارده ایم، در درون زمانبند هر گره قرار گرفته و به صورت دوره ای عملیات خود را انجام می دهد. در این روش در هر دوره اطلاعات بار همسایه های گره به همراه تاخیر زمانی ارتباط با آنها جمع آوری می شود. سپس میانگین بار گره های همسایه محاسبه می شود، در صورتیکه گره بیش از میانگین همسایه هایش بار داشته باشد، خود را پر بار نامیده و سعی در موازنه بار خود در سطح همسایه ها می کند. الگوریتم در این زمان با استفاده از روشی به نام مسیریابی مجازی، سودمند ترین همسایه خود را

به عنوان مقصد انتخاب و اضافه بار خود را به آن ارسال می کند. باید توجه کرد که در این روش اطلاعات رد و بدل شده در درون بسته های ارتباطی که بطور دوره ای بین عاملهای همسایه در ARMS مبادله می شود قرار می گیرد و بنابراین هیچ سرباری برای سیستم ندارد. عیب اصلی این روش این است که چون دامنه دید در آن محدود است، ممکن است یک گره بهترین تصمیم را پیدا نکند همچنین سرعت همگرایی نیز در این روش کم است.

- در پایینترین سطح، الگوریتمی بر پایه الگوریتمهای ژنتیک قرار دارد که بار تخصیص داده شده به هر گره را به صورت بهینه بر روی پردازنده های آن گره پخش می کند.

در واقع MLBM عملکرد همزمان این سطوح در قالب سیستم مدیریتی ARMS است.

۱-٦- نمای کلی پایان نامه

رئوس مطالب این پایان نامه به این شکل طبقه بندی شده که در مقدمه به طور کلی در مورد محیط گرید، تاریخچه، خصوصیات گرید، اجزاء گرید، سیستمهای مدیریت منابع، راهکارهای عاملگرا، سیستم مدیریتی ARMS و نقاط ضعف آن، لزوم موازنه بار در محیط گرید و انواع روشهای ممکن برای آن، و در نهایت خلاصه ای از روش ارائه شده برای موازنه بار در ARMS آورده شده.

در فصل ۲، ابتدا تعاریف گرید و اهداف و ویژگیهایی که از آن انتظار می رود آورده شده. در قسمت بعد انواع محیطهای معروف و مطرح گرید مورد بررسی دقیق قرار گرفته اند. در ادامه مبانی و اصول سیستمهای مدیریت منابع مطرح در سیستمهای توزیع شده محاسباتی و سپس در گریدهای محاسباتی مورد بحث قرار گرفته و با هم مقایسه شده اند. در قسمت بعد مدلهای پایهٔ موجود برای مدیریت منابع در محیط گرید به همراه مزایا و معایب هر یک مورد بحث قرار گرفته اند. در مرحله بعد بررسی کاملی در مورد عاملها، سیستمهای چند عامله، کاربردهای آنها مورد بحث قرار گرفته اند. در مرحله بعد بر روی نحوه کار و ساختار یک سیستم مدیریت منابع عاملگرا به نام ARMS تمرکز کرده و نکات قوت و ضعف آن را برشمرده ایم.

در فصل ۳ به بررسی کامل روشهای موازنه بار پرداخته ایم. در ابتدا تعریفهایی را که تا کنون برای موازنه بار در محیطهای توزیع شده و گرید بیان شده اند را بطور کامل ارائه داده ایم. ویژگیهایی که یک الگوریتم موازنه بار به ویژه در محیط گرید باید دارا باشد را برشمردیم و سپس روشهای موجود را بر اساس همان ویژگیها دسته بندی کرده ایم . در قسمت بعد مشخص کرده ایم که برای انجام روالهای موازنه بار چه مراحلی باید انجام شود. سپس به بررسی رویکردهای موجود در موازنه بار پرداخته ایم در این میان به الگوریتمهای مبتنی بر گروه مورچه ها، روشهای مبتنی بر الگوریتمهای ژنتیک و همچنین روشهای سطح زمانبند که یا در ARMS پیاده سازی شده اند و یا در روش ارائه شده این پایان نامه بکار رفته اند، اهمیت بیشتری داده ایم.

در فصل ٤ در ابتدا مقدمه کوتاهی درباره گرید، مدل ARMS و ضعفهای آن در موازنه بار گره ها آورده شده. در قسمت بعد ویژگیهایی که الگوریتم موازنه بار ارائه شده، بنا به ویژگیهای ARMS باید داشته باشد ارائه شده است. سپس لایه های مدل چند سطحی MLBM (مدل ارائه شده) به صورت کلی مورد بررسی قرار گرفته است. در مرحله بعد لایه مربوط به

الگوریتم مورچه های هوشمند بحث شده و به صورت ریاضی اثبات کرده ایم که کارایی بهتری نسبت به مدل اولیه ارائه می دهد. سپس لایه مربوط به مدل همسایگی به صورت ریز و با جزئیات کامل مورد بحث قرار گرفته است. سپس به بررسی تعامل این دو روش در قالب مدل MLBM پرداخته ایم. با تکمیل بیان مدل ارائه شده محیط شبیه سازی و آزمایشهای لازم برای نشان دادن کارایی سیستم را معرفی کرده ایم. در مرحله بعد به معرفی تک تک آزمایشها و بحث و بررسی بر روی نتایج آنها پرداخته ایم.

در فصل ۵ ضمن نتیجه گیری کارهای انجام شده، سعی شده زمینه هایی را که این کار تحقیقی در گامهای بعد و در جهت عملیاتی شدن نیاز دارد مورد بررسی قرار گیرد.

فصل ۲

سیستمهای مدیریت منابع در گرید

۲-۱- مقدمه

یک گرید، یک کامپیوتر مجازی بزرگ است که از تعداد زیادی از کامپیوترهای ناهمگون که بر سر به اشتراک گذاری منابع خود با هم به توافق رسیده اند، بوجود می آید. گرید یک نمونه از شبکه‌های محاسباتی توزیع شده با مقیاس بسیار وسیع است که می‌تواند در مقیاس اینترنت رشد کرده و در بین چندین سازمان و دامنه‌های مدیریتی متفاوت فعالیت نماید. پیدایش برنامه‌های کاربردی و بسیار جدی در این زمینه، نیاز به ارائه مکانیزمهای جدید مدیریت داده و منابع را در چنین محیط‌هایی بسیار حائز اهمیت ساخته است. طراحی معماری مناسب برای گرید از چندین منظر حائز اهمیت است[۳۵،٤٦]. گرید باید بتواند

الف) قابلیت انطباق،توسعه پذیری و مقیاس پذیری داشته باشد.

ب) امکان ارتباط سیستم‌های موجود تحت مدیریت‌های متفاوت را با استانداردهای ناهمگون فراهم آورد.

ج) امکان اختصاص مشترک منابع را فراهم آورد.

د) کیفیت خدمات را تضمین نماید.

هـ) محدودیتهای هزینه‌های محاسباتی را در هنگام اختصاص منابع در نظر بگیرد.

و) از منابع به طور بهینه و متوازنی استفاده نماید.

ز) استحکام وپایداری لازم در برابر بروز خطاهای احتمالی - که به علت تنوع و گستردگی منابع در گرید شایع نیز می باشد- را داشته باشد.

سیستم مدیریت منابع، مؤلفه اصلی سیستمهای گرید می‌باشد. پروژه‌های متعددی به منظور توسعه گرید اجرا شده‌اند که معماری ها و سرویس‌های گوناگونی را در این عرصه طراحی و پیاده سازی کرده‌اند. در این بخش سعی شده است تا مدلهای گوناگون پیشنهاد شده برای مدیریت منابع در این گریدها را معرفی کنیم.

سیستم مدیریت منابع[۱] مجموعه‌ای از منابعی را که در حال حاضر در اختیار دارد، مدیریت می نماید. به عنوان مثال زمان‌بندی پردازشگرها، پهنای باند شبکه و وسائل جانبی ذخیره ساز اطلاعات از جمله‌ی این وظایف می‌باشند. در گرید، منابع متولیان مختلفی دارند، بنابراین سیستم مدیریتی مورد استفاده باید مورد توافق تمامی صاحبان منابع باشد. به علت بدون مرز بودن و گستردگی گرید، استفاده از روشهای مدیریتی مرکزی، کارآیی و قیاس پذیری سیستم را به شدت مورد تهدید قرار داده و سیستم را آسیب پذیر و نامطمئن می نماید. برای جلوگیری از این مشکلات به طور کلی بهتر است از رویکردهای مدیریتی توزیع شده استفاده کرد. به عنوان مثال می‌توان چندین سیستم مدیریت منابع را مورد استفاده قرار داده و مطابق با یک مجموعه پروتکل‌های توافق شده تشکیل یک فدراسیون از سیستمهای مدیریت منابع را داد. سیستمهای مدیریت منابع می‌توانند با استفاده از این پروتکل‌ها با هم تبادل اطلاعات نمایند.

برنامه‌های کاربردی می‌توانند به صورت مستقیم و یا غیر مستقیم از گرید درخواست منابع نمایند. درخواست‌های ارسال شده به گرید تحت عنوان وظایف، شناخته می‌شوند. بسته به نوع وظیفه، برنامه‌های کاربردی می‌توانند سطح مورد انتظار کیفیت خود را معرفی کرده و یا از مدل بهترین تلاش[۲] پیروی نمایند. سیستم مدیریت منابع موظف است پارامترهای کیفیت سرویس را در بالاترین مقدار ممکن برآورده کند. در عمل، هر سیستم مدیریت منابع، وظایف متفاوتی را با مدلهای زمانبندی گوناگونی مدیریت می کند.

---

[۱] Resource Management System(RMS)

[۲] Best effort

به عنوان مثال، برخی از وظایف، باید پردازش خود را تا یک موعد زمانی خاص حتما انجام دهند و تضمین این سرویس را از مدیر منابع گرید درخواست کنند، در حالی که بعضی دیگر ممکن است چنین درخواستی را مطرح نکرده و رفتار پردازشهای دسته ای را از خود نشان دهند. همانطور که از توضیحات قبلی مشخص است مسأله زمانبندی درون گرید یک مسأله با پارامترها و محدودیت های گوناگون و متعدد است. در این فصل سعی داریم تا اجزاء حیاتی یک سیستم مدیریت منابع را معرفی کنیم. اجزاء گوناگون معرفی شده بر طبق مدلهای مرجع موجود طبقه بندی شده اند.

۲-۲- گرید و محاسبات گریدی

هدف از محاسبات گریدی، ترسیم یک کامپیوتر مجازی بزرگ، قدرتمند و خود مدیریت شونده است که از اجتماع کامپیوترهایی که هر یک منبع (منابعی) را در اختیار گذاشته اند تشکیل شده است. آنچه تا کنون در اینترنت انجام شده استاندارد سازی ارتباطات داده ها میان سیستم های مختلف و ناهمگون بوده است و اکنون در گرید استاندارد سازی اشتراک منابع در دستور کار محققان قرار گرفته است. به عبارتی دیگر پروتکلهای گرید در پی ایجاد سیستم عاملی برای گرید می باشند.

تاریخچه گرید به اواسط دهه گره و با پروژه ای به نام FAFNER [٦٣] شروع گردید که در آن از سرویس دهنده های وب به عنوان گره های گرید و به منظور ساختن کلیدهای RSA استفاده می شد. ایده های اساسی FAFNER مورد قبول واقع شده و بر اساس آنها پروژه هایی همچون Distributed.net [۱۳] و SETI@home [۸] به وجود آمدند. در سیستمهای گرید اولیه مشکلات جدی مربوط به ناهمگونی گره ها، عدم قیاس پذیری و تطبیق پذیری وجود داشت که امروزه با حل آنها توسط سیستمهای مدیریت منابع مختص گرید همچون GloBus [۲۲]،Condor [۵۹]، Legion [۱] و Jini [٤٨] زیر ساخت مناسبی برای گرید سراسری به وجود آمده است. در ضمیمه الف فهرست کاملی از پروژه های در حال انجام گرید آورده شده است. محققین بر این باورند که با رفع نقائص سیستمهای فعلی، در آینده استفاده از گریدهای محاسباتی به سادگی و پرکاربردی نیروی برق خواهد بود[۹۵]. در اینصورت شما اجرای برنامه های خود را به گرید می سپارید و مطابق با میزان استفاده، هزینه خواهید پرداخت.

۲-۲-۱- تعاریف مختلف گرید

در این قسمت به چند تعریف برای گرید از دید دانشمندان این رشته می پردازیم.

یان فاستر مبدع اصلی نام گرید و بنیان گذار گلوباس، گرید را به صورت زیر تعریف می کند[۳۰] :

" فناوری گرید به دنبال بوجود آوردن امکان اشتراک منابع در مقیاس وسیع و به صورت کنترل شده و انعطاف پذیر است و بدین جهت به دنبال ایجاد پروتکلها، سرویسها و بسته های نرم افزاری است."

در پروژه GridStart که یکی از پروژه های گرید در سطح اروپاست، گرید به صورت زیر تعریف شده است:

" گرید گام بعدی اینترنت است که در آن پهنای باند بالا، محاسبات پرسرعت، حسگرهای هوشمند و پایگاه داده های عظیم به صورت انباری است که پنهان از دید کاربران، مدیریت شده و بنابراین به راحتی می تواند در اختیار همگان قرار گیرد. [۲۱]"

در شرکت IBM که یکی از پیشگامان در زمینه گرید می باشد گرید چنین تعریف می شود:

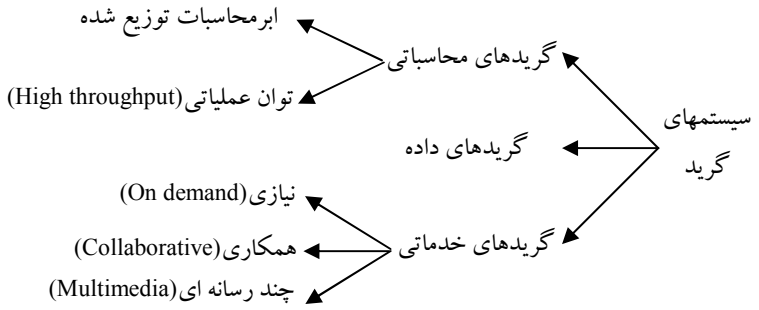
" محاسبات گرید، همان محاسبات توزیع شده است که پا به مرحله تکامل نهاده و هدف آن ارائه کامپیوتر مجازی بزرگ و خود مدیریت شونده است [۵۲]"

درنهایت در پروژه گلوباس که استانداردترین پروژه گرید است، گرید به صورت زیر تعریف می شود. این تعریف به عنوان تعریف استاندارد گرید پذیرفته شده است.

" گرید به معنی زیربنایی برای یکپارچه سازی و استفاده مشترک از کامپیوترها، شبکه ها، پایگاه داده ها و تجهیزات عملی است که به سازمانهای مختلف تعلق دارد وتوسط خود آنها سیاست گذاری و مدیریت می شود. [۲۲]"

۲-۲-۲- انواع محیطهای گرید

این بخش اهداف طراحی و برنامه‌های کاربردی مورد نظر جهت توسعه معماری گرید را معرفی می نماید. در این بخش با سه رویکرد به طراحی معماری گرید می‌پردازیم : ۱. بهبود کارایی برنامه های کاربردی ۲. دسترسی به داده‌ها ۳. بهبود خدمات موجود در گرید. با این دیدگاه سیستم‌های گرید در طبقه بندی که در شکل ۲-۱ دیده می‌شود، قرار می‌گیرند.



شکل۲-۱. طبقه بندی انواع گرید .

مدل گرید محاسباتی، زمانی مورد استفاده قرار می‌گیرد که مجموعه توانایی محاسباتی کامپیوترهای موجود برای اجرای یک برنامه کاربردی، از تک‌تک کامپیوترهای تشکیل دهنده گرید بیشتر باشد. بسته به نوع بهره گیری از این توانایی، این نوع گرید به دو دسته جزئی‌تر تقسیم می‌شود که به ترتیب ابرمحاسبات توزیع شده و گریدهای با توان عملیاتی بالا نامیده می‌شوند. گریدهای ابر محاسبه‌گر، معمولا یک برنامه‌ی کاربردی را به صورت موازی بر روی چندین ماشین اجرا کرده و زمان اجرای آن را کاهش می‌دهند. این نوع گرید برای برنامه هایی همچون مدلسازی وضع جوی و شبیه سازهای صنعتی ونظامی که محاسبات بسیار زیادی نیاز دارند مناسب است. گریدهای با عملکرد بالا بر خلاف حالت قبل، به دنبال انجام تعداد کارهای بیشتر در واحد زمان هستند [۷۲،۲،۱۰].

گریدهای داده برای سیستم‌هایی طراحی شده‌اند که زیر ساختار مناسبی را برای تولید اطلاعات از داده‌های توزیع شده موجود در سطح شبکه مانند کتابخانه‌های دیجیتال و یا انباره‌های داده فراهم می‌کنند. گریدهای محاسباتی هم امکانات خاصی را برای مدیریت داده‌ها فراهم می نمایند، ولی گریدهای داده‌ای دارای زیر ساختهای مناسبی برای مدیریت ذخیره سازی و دسترسی به داده‌ها هستند. در یک گرید محاسباتی غالباً برنامه‌های کاربردی، مدلهای دسترسی و ذخیره داده مخصوص به خود را پیاده سازی می کنند و به صورت متمرکز از یک مدل استفاده نمی‌کنند. پروژه‌های عمده‌ای

در زمینه گریدهای داده‌ای از جمله [۳۱] Globus و [۸٦] European DataGrid Project در حال اجرا هستند که هدف آنها طراحی سازمانهای وسیع مدیریت اطلاعات و دسترسی به داده‌ها می باشد.
از انواع دیگر گرید، گریدهای خدماتی می‌باشند. این نوع گریدها، خدماتی را که هیچ ماشینی به تنهایی ارائه نمی کند به استفاده کنندگان گرید ارائه می‌کنند. گریدها در این دسته به گروههای گریدهای همکار،نیازی و چند رسانه‌ای تقسیم می‌شوند. در یک گرید همکار برنامه‌های کاربردی و کاربران درخواست کننده سرویس، در یک فضای مجازی به هم متصل شده و از طریق این فضای مجازی کاربران می توانند به صورت بلادرنگ با منابع محاوره داشته باشند. گریدهای نیازی به صورت پویا منابع موجود گوناگون را برای اجرای یک برنامه کاربردی خاص متحد کرده تا یک سرویس جدید را ارائه کنند، به عنوان مثالی از گریدهای نیازی، یک محیط شبیه سازی را می توان در نظر گرفت که در آن آزمایش کننده می خواهد صحت جوابهای خود را با تعداد متفاوتی از کامپیوترهای محاسبه گر در هر لحظه بررسی کند، امکان افزایش تعداد ماشین‌های محاسبه‌گر براساس درخواست کاربر می تواند با استفاده از یک گرید نیازی فراهم شود. گریدهای چند رسانه‌ای زیرساخت مناسب برای اجرای برنامه‌های چندرسانه‌ای بلادرنگ را فراهم می نمایند. این عمل نیاز به تضمین کیفیت سرویس بر روی ماشین‌های متعدد را دارد.
اغلب فعالیت‌های تحقیقاتی در زمینه سیستم‌های گرید در یکی از دسته‌های فوق قرار می‌گیرند. هدف بلند مدت تمامی محققین، ایجاد مدلی از گرید است که تمامی مدلهای فوق را در بر گیرد.

۲-۳- سیستم مدیریت منابع

در این بخش به بررسی مدلهای موجود سیستمهای مدیریت منابع می‌پردازیم و ساختار معماریهای موجود را بررسی می‌کنیم. به منظور خلاصه سازی بحث‌های مطرح در این زمینه، تنها اجزا و عملکردهای اصلی سیستمهای مدیریت منابع را در نظر خواهیم گرفت.
در گرید، منبع یک موجودیت قابل بازیافت و استفاده مجدد بوده که می تواند نیازهای یک درخواست و یا وظیفه را برآورده سازد. این منبع می‌تواند یک ماشین، شبکه و یا حتی نرم افزار باشد. فراهم کننده منبع عاملی است که منبع مورد نظر را سیاست گذاری و کنترل می نماید و به طور مشابه مصرف کننده منبع، عاملی است که از منبع استفاده می‌کند. در ضمیمه ب لیست کاملی از سیستمهای مدیریت منابع گرید آورده شده است. بررسی کاملتر در این زمینه در [٤٧] موجود است.

۲-۳-۱- مقایسه مشخصه های سیستمهای مدیریت منابع در سیستمهای توزیع شده و گرید

با توجه به تعاریف انجام شده، یک سیستم مدیریت منابع در یک سیستم توزیع شده سرویسی است که مدیریت حجم زیادی از منابع موجود در شبکه را به عهده گرفته و با توجه به سیاستهای مالک منبع آنها را به طور بهینه به درخواستها نسبت می دهد.
مساله مدیریت منابع در گرید علیرغم شباهتهای زیاد با سیستمهای توزیع شده، از دو جهت عمده که در زیر به آنها اشاره می شود با مدیریت منابع درسیستمهای توزیع شده متفاوت است.
الف)نیاز به توجه در مباحث جدیدی همچون ناهمگونی، قیاس پذیری، توسعه پذیری، انطباق پذیری، استقلال، کیفیت خدمات و اختصاص همزمان منابع،در گرید که این مباحث در سیستمهای توزیع شده مطرح نیستند. ب) به علت شرایط

خاص محیط گرید، بسیاری از موارد مشترک با سیستمهای توزیع شده همچون پاسخگویی، تحمل خطا، پایداری و استفاده متوازن از منابع، در گرید نیازمند راهکارهای متفاوتی به نسبت سیستمهای توزیع شده می باشند. نیازهای کیفیت سرویس، نمونه ای از این مسائل است. سیستمهای توزیع شده سنتی عمدتاً یک دامنه مدیریت شده محدود را شامل می‌شوند و به همین دلیل وظایف ارسال شده از سوی مشتریان، در همان محدوده مدیریتی اجرا می‌شوند. به همین منظور کیفیت سرویس در اینگونه محیط‌ها با توجه به اولویت وظایف به نحو مطلوبی صورت می‌گیرد. از سوی دیگر در گرید، به خاطر وجود درخواستها از دامنه‌های مدیریتی متفاوت و سطح تغییرات زیاد(عدم وجود مرز مشخص) و همچنین به دلیل اینکه اجرای یک تقاضا بستگی به نوع اجازه‌های دسترسی، نوع منابع مورد نیاز و منابع در دسترس برای تضمین سطح کیفیت سرویس دارد، فراهم کردن کیفیت سرویس بسیار دشوار می باشد.

۲-۳-۲- مدلهای پایه برای مدیریت منابع

ساختار کلی یک سیستم مدیریت منابع می‌تواند بر اساس خصوصیات مختلف آن بررسی شود. تفاوتهای ساختاری در پیاده سازیهای متفاوت از آنجا ناشی می‌شود که معماریهای گوناگون به دلیل وجود نیازهای متفاوتی طراحی شده‌اند. بدلیل اینکه گرید به طور منطقی منابع مختلفی را گرد هم می آورد که از گروه ها و نهادهای متفاوتی می باشند، انتخاب یک روش مناسب مدیریت منابع نقش زیادی در میزان موفقیت آن دارد. روشهای مختلفی برای مدیریت منابع وجود دارند که می توان آنها را به سه دسته اصلی زیر تقسیم بندی کرد :

- مدل سلسله مراتبی[۱]
- مدل مستقل از مالک[۲]
- مدل اقتصادی/فروش قدرت محاسبات[۳]

در جدول۲-۱ دسته بندی سیستمهای مختلف بر این اساس بیان شده است. باید توجه کرد که روش دوم عملا با توجه به سخت افزارهای امروزی هنوز قابل پیاده سازی نیست [۷۳].

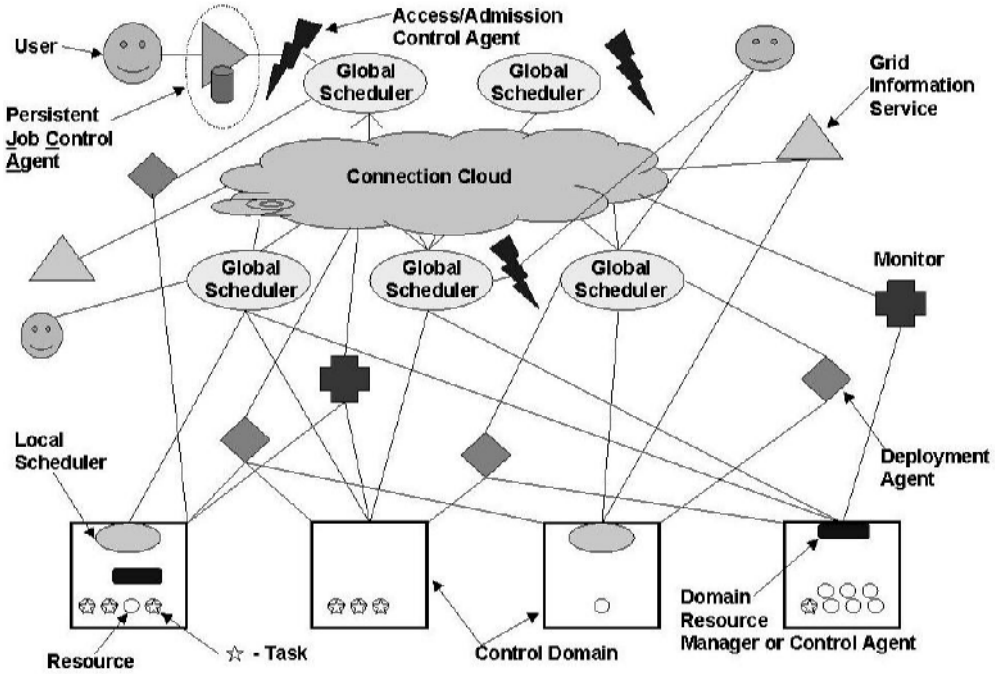
مدل	توضیح	سیستم
سلسله مراتبی	در اکثر رویکردهای فعلی استفاده شده است.	Globus, Legion, Ninf, NetSolve.
مستقل از مالک	بر اساس منطق عرضه و تقاضا عمل می کند البته هنوز مسائل بسیاری در آن مطرح است.	هنوز مورد عملی وجود ندارد.
اقتصادی	از مدلهای اقتصادی موجود و خواص سیستمهای سلسله مراتبی و مستقل از مالک در کشف ومعرفی منابع استفاده می کند.	Nimrod/G, JaWS, Myriposa, JavaMarket

جدول ۲-۱. مدلهای مدیریت منابع.

[۱] Hierarchical Model
[۲] Abstract owner Model
[۳] Computation Market / Economy Model

الف) مدیریت منابع سلسله مراتبـی

این روش حاصل دومین نشست انجمن گرید بود. اجزاء اصلی این معماری که در شکل۲-۲ نشان داده شده، اجزاء غیر فعال[۱] و فعال[۲] هستند.



شکل۲-۲. مدل سلسله مراتبی [٤٧]

اجزاء غیر فعال :

- شامل منابعی هستند که پاره ای از اوقات می توانند استفاده شوند.علاوه بر اینکه این منابع ممکن است تجدیدشدنی یا تجدید نشدنی باشند، می توانند به صورت مشترک یا انحصاری نیز باشند و همچنین می توانند دارای نام مستقل[۳] بوده و یا به صورت پارامتری توضیح داده شوند. به عنوان مثال پهنای باند شبکه، فضای دیسک و زمان پردازنده را می توان نام برد.
- وظایف[٤]: مصرف کننده منابع بوده و شامل پردازشهای مرسوم (عادی) و پردازشهای غیر مرسوم مثل پیغام ها هستند.
- کارها[٥]: موجودیتهای سلسله مراتبی هستند که ممکن است ساختاری بازگشتی داشته باشند. یعنی کارها می توانند متشکل از زیر کارها یا وظایف باشند. زیر کارها نیز می توانند شامل زیرکارهای دیگری باشند. در نهایت وظیفه ها برگهای این سلسله مراتب هستند.
- زمانبندی ها[٦]: در واقع بیان کننده نحوه نگاشت وظایف به منابع در طول زمان می باشند. باید توجه کرد که وظایف هستند که به منابع نگاشت می شوند نه کارها. در واقع کارها به عنوان ظرفی برای وظایف عمل می کنند.

---

[۱] Passive
[۲] Active
[۳] Explicit Name
[٤] Tasks
[٥] Jobs
[٦] Schedules

اجزاء فعال :

- زمانبندها[۱]: یک یا چند روش زمانبندی برای کارها مشخص می کنند. واحد زمانبندی، کار است به این معنی که زمانبند سعی می کند تمام وظایف موجود در یک کار را یکجا به یک منبع (قسمت) نگاشت کند. بنابراین کارها (و نه وظایف) به زمانبند منتقل می شوند.
- سرویسهای اطلاعاتی[۲]: به عنوان پایگاه داده ای برای توضیح اقلام مورد نیاز سیستم مدیریت منابع، از قبیل منابع، کارها، زمانبندها، عاملها و موارد دیگر می باشند. روش دسترسی که در این قسمت مورد استفاده قرار می گیرد می تواند LDAP، پایگاه داده تجاری یا هر چیز دیگری باشد.
- عاملهای کنترل دامنه[۳]: این عاملها وظیفه ارجاع به منبع برای استفاده از آن منبع را دارند. منظور از دامنه کنترل، مجموعه منابعی است که توسط یک عامل کنترل می شوند. این عاملها عملیات رزرواسیون را انجام می دهند و با زمانبندها فرق دارند. اما یک عامل کنترل دامنه ممکن است شامل زمانبندهای داخلی نیز باشد. یک عامل از این نوع می تواند یکسری اطلاعات از حالت منابع را داشته باشد. این اطلاعات می تواند از طریق ثبت در یک سرویس اطلاعات و یا از طریق پرس و جوی مستقیم بدست آید. زمانبند Maui و GRAM در Globus [۳۲] وهمچنین اشیاء میزبان[٤] در Legion [۵۱] نمونه هایی از این عاملها هستند. زمانبندهایی وجود دارند که درون عاملهای کنترل دامنه قرار ندارند و به آنها ابرزمانبند[۵] گفته می شود.
- عاملهای به کارگیری[٦] : نقش پیاده سازی زمانبندی را دارند. این کار توسط مذاکره باعاملهای کنترل دامنه برای بدست آوردن منبع و اجرای برنامه بر روی آن انجام می شود.
- عاملهای کنترل پذیرش[۷]: مشخص می نماید که یک سیستم می تواند هنوز کار قبول کند یا نه. به عبارت دیگر همان وظیفه زمانبند دراز مدت را در سطح گرید انجام می دهد.
- ناظرها[۸]: عملیات پیگیری و نظارت بر اجرا را انجام می دهند. مانیتورها عملیات خود را از طریق وظایفی که کار را تشکیل داده اند و همچنین عامل کنترل دامنه آنها بدست می آورد. بر اساس این اطلاعات ناظر ممکن است فراخوانیهایی از عاملهای کنترل کار انجام دهد تا بتواند باعث ایجاد موقعیت جدیدی (از لحاظ زمانبندی) برای کار شود.
- عاملهای کنترل کار[۹]: وظیفه هدایت کار درون سیستم را دارند که هم می توانند به عنوان وکیلی برای کاربر و هم یک نقطه کنترل ثابت (پایدار) برای کار باشند. به عبارت دیگر وظیفه هماهنگی بین اجزاء مختلف سیستم مدیریت منابع را به عهده دارند. به عنوان مثال به صورت هماهنگ کننده ای بین مانیتور و زمانبندها عمل می کنند.

موارد ذکر شده در واقع یک تفکیک منطقی بین اجزاء سیستم است و در یک سیستم خاص ممکن است چندین قسمت از قسمتهای ذکر شده با هم ترکیب شوند. اگر دقت شود در بحث قبل از روش صف بندی

---

[۱] Schedulers
[۲] Information service
[۳] Domain Control Agents
[٤] Host Object
[۵] Meta scheduling
[٦] Deployment Agent
[۷] Admission Control Agent
[۸] Monitors
[۹] Job Control Agents

چیزی گفته نشد چون در واقع صف بندی به نوعی بیان کننده همگونی منبع می باشد که این امر در گرید وجود ندارد. در واقع صفها را در این سیستم درون عاملهای کنترل دامنه باید قرار داد.

محاوره اجزاء :

همانطور که در شکل۲ دیده می شود کاربر، کار(تقاضا) خود را ابتدا به یک عامل کنترل کار(JCA) می فرستد. که این عامل به عنوان وکیلی برای کاربر عمل کرده و عامل پذیرش (AA) را فراخوانی می کند. این عامل همانند زمانبند دراز مدت ابتدا به منابع مورد نیاز کار توجه کرده (ممکن است درانجام این کار با سرویسهای اطلاعاتی همکاری کند) و مشخص می کند که آیا می توان این کار را به مجموعه کارهای در حال اجرا بر روی گرید اضافه کرد یا نه. سپس عامل پذیرش (AA) کار را به یک زمانبند فرستاده و در این مرحله عملیات پیدا کردن منبع مورد نظر به کمک سیستمهای اطلاعاتی انجام می شود. در مرحله بعد با عاملهای کنترل دامنه ارتباط برقرار شده و وضیعت منابع چک می شوند. سپس زمانبند یکسری نگاشتها (ترتیبها) را مشخص کرده و آنها را به عاملهای بکارگیری (DA) می فرستد. این عاملها با عاملهای کنترل دامنه تماس گرفته و عمل رزرواسیون برای آن منبع خاص را انجام می دهند و در زمان مناسب JCA می تواند اجرای واقعی را شروع کند. از مانیتورها می توان برای نظارت بر نحوه اجرا استفاده کرد و در صورت نیاز عملیات زمانبندی مجدد برای رسیدن به بهره وری مناسب انجام شود.

روالی که ذکر شد یک روند ممکن برای هماهنگی اجزای مختلف سیستم است و الزاما در همه پیاده سازیها این ترتیب و تفکیک وجود ندارد.

ب) مدل مستقل از مالک[۱]

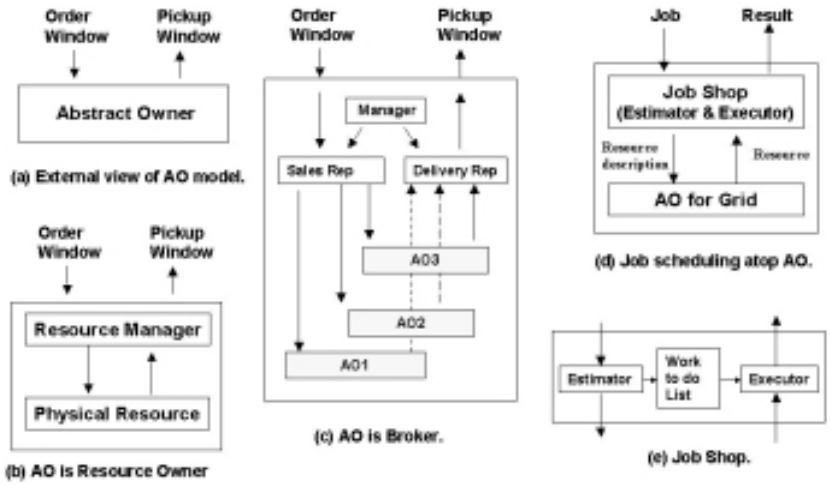
وقتی یک تماس تلفنی گرفته می شود منابع مورد استفاده در این ارتباط متعلق به چه کسی است؟ یا صاحب اینترنت چه کسی است؟ اینها سوالاتی هستند که البته کاربران نهایی جواب آنرا نمی دانند ودانستن آن هم برای آنان مهم نیست. کاربران فقط سرویس می خواهند یعنی کاربران به دنبال این هستند که بتوانند از منابع با شرایط خوب استفاده کنند. به عبارت دیگر فقط احتیاج به ابزارهایی برای بکارگیری منابع، هزینه سنجی و همچنین برای پرداخت هزینه داریم. آنچه کاربران با آن تماس می گیرند در اغلب مواقع مالک اصلی منبع نیست بلکه واسطه هایی برای ارتباط با منابع است. خود این واسطه ها ممکن است مستقیما و یا حتی از طریق واسطه های دیگر به صاحبان اصلی منابع متصل باشند و این امرهیچ اهمیتی برای کاربر نهایی ندارد. مثلا نقش ISP ها به این گونه است. در گرید نیز روش کار به همین صورت می باشد. کاربر گرید احتیاجی به دانستن اینکه چه کسی مالک گرید است ندارد، پس در اینجا نیز از تکنیکی مشابه آنچه تاکنون استفاده می شده استفاده می کنیم، یعنی استفاده یک یا چند لایه تجرید مالکیت که به آنها Ao گفته می شود. بنابراین از این پس کاربر می تواند مالک یک منبع را یک Ao فرض کند. نماینده کاربر نهایی، برنامه کاربر است که به آن مشتری[۲] گفته می شود. پس در واقع آنچه اتفاق می افتد مذاکره و تراکنشی است بین مشتری یک منبع و Ao متناظر آن منبع. نحوه مشخص شدن هزینه استفاده از منبع نیز می تواند به طرق گوناگونی انجام شود، مثلا به صورت یکنواخت باشد و یا این که بنا به مدت زمان نیاز به منبع، هزینه آن تغییر کند و یا حتی می تواند به

---

[۱] Abstract Owner Model
[۲] Client

صورت توافقی بین Ao و مشتری صورت گیرد. البته این هنوز در حد یک طرح است و باید تغییرات و بهینه سازیهای زیادی بر روی آن صورت گیرد تا به صورت عملی در آید.

ساختار کلی مالک مجازی[۱] : در ساده ترین شکل یک Ao از دید بیرونی مثل یک رستوران خیابانی عمل می کند. یعنی از یک پنجره تقاضا داده می شود و از پنجره دیگر غذا تحویل داده می شود! به عبارت دیگر برای دسترسی به منبع از یک Ao که صاحب آن منبع است، مشتری (که می تواند یک Ao دیگر باشد) از طریق پنجره سفارش، مذاکره را در مورد قیمت و اینکه چه زمانی منبع را در اختیار خواهد گذاشت شروع می کند. اگر نتایج با نیازهای کاربر تطابق نداشت می تواند مذاکره را قطع کرده ویا اینکه سفارش سرویس دهد. پس از سفارش، Ao منابع را به مشتری تحویل می دهد. البته تحویل می تواند هم به صورت سرکشی و هم به صورت وقفه به مشتری داده شود. حتی در زمانیکه سفارش داده شده اما هنوز منبع تخصیص نیافته هم می توان عملیات را از طریق پنجره تقاضا قطع کرد. فقط باید به این تفاوت توجه کرد که در اینجا تقاضا ممکن است به صورت راه دور باشد پس باید مکانیزمهایی برای فرستادن پارامترها وجود داشته باشد که برای این منظور می توان از CORBA یا هر روش دیگری برای فراخوانی استفاده کرد.



شکل۲-۳. مدل مستقل از مالک [٤٧].

منابع را می توان به صورت اشیایی که دارای یکسری صفات و روشها هستند بیان کرد. در واقع صفات باعث تنظیم شدن منبع به صورت دلخواه شده و توابع عملیات متفاوتی را برای کاربر انجام می دهند. البته صفات را فقط در هنگام تقاضای منبع می توان تنظیم کرد ودرحین در اختیار داشتن منبع این کار ممکن نیست و فقط میتوان آنها را خواند.

ساختار بیان شده به گونه ای است که امکان لانه ای شدن[۲] را فراهم می آورد، از لحاظ داخلی هم AO ها به گونه ای هستند که می تواند واسطه باشد و یا مالک و یا ترکیبی از هر دو. یک شی که فقط مالک (مالک اصلی) است ساختار بسیار ساده ای دارد. در چنین سیستمی"مدیر" توانایی مذاکره و زمانبندی و تحویل منبع را

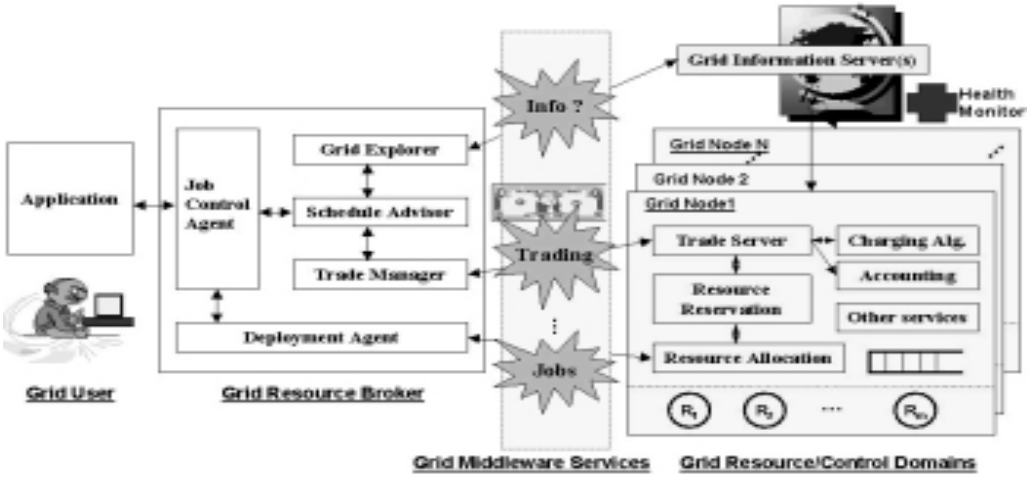
[۱] Abstract Owner
[۲] Nested

دارد. این ساختار برای واسطه های سطوح بالاتر پیچیده تر است. قسمتهای مختلف شکل ۲-۳ این مسائل را به خوبی نشان می دهد.

با این حال نکات زیادی در مورد این روش هنوز بدون جواب مانده که باید در مرحله عمل به آن پاسخ گفت.

ج) مدل اقتصادی[۱]

منابع موجود در گرید به لحاظ جغرافیایی و مالکیت گستردگی زیادی دارند و متعلق به نهادهای مختلفی در سراسر جهان می باشند، که هر نهاد سیاستهای خود را برای ارائه خدمات وهزینه های مربوط به ارائه منابعش را دارد. بنابراین یک سیستم مدیریت منابع بایستی امکاناتی برای پشتیبانی از مسائل اقتصادی را نیز داشته باشد. مسائل اقتصادی باعث انگیزه بیشتری در تشریک منابع توسط صاحبان منابع می شود. در مسائل اقتصادی همیشه کاربر منبع به دنبال کم کردن هزینه و مالک منبع به دنبال افزایش سود است. پس سیستم مدیریت منابع باید به گونه ای باشد که نیاز هر دو طرف را برآورده نماید. مثلا در طرف مالک باید امکاناتی همچون نحوه استفاده، قیمت به نسبت زمان و قیمت به نسبت کاربران مختلف وجود داشته باشد. سیستمهایی همچون Mariposa[۶۲] و Nimrod/G [۷۲] و JaWs [۸۱] چنین طرحهایی را ارائه داده اند و سایر سیستمهای کنونی نیز درحال حرکت به این سمت هستند. این مدل مدیریت منابع ویژگیهای Ao و مدل سلسله مراتبی را دارا می باشد یکی از این نمونه مدیریت منابع در شکل۲-٤ آورده شده است.



شکل۲-٤. مدل مدیریتی اقتصادی[٤۷].

اجزاء اصلی یک سیستم مدیریت منابع به این شیوه عبارتند از :

- برنامه کاربر:که می تواند سریال، موازی، پارامتری و یا همکار با چند پردازش دیگر باشد.
- واسطه منبع گرید (GRB)[۲]. به عنوان واسط ای بین کاربر ومنابع موجود در گرید قرار گرفته و وظایفی همچون پیدا کردن منبع مناسب، الحاق برنامه به منبع و شروع برنامه بر روی آن را به عهده دارد. به عبارت کلی تر وظیفه دارد که یک دید یکسان و کاملا شفاف از گرید برای کاربر به وجود آورد.

---

[۱] Economy / Market Model
[۲] Grid Resource Broker

- میان افزار گرید[۱]: وظیفه ارائه سرویسهایی که باعث هماهنگی بین کاربر و منبع می شود را از طریق واسطه منبع به عهده دارد. وظیفه ارائه سرویسهای اصلی مثل مدیریت پردازش راه دور، تخصیص همزمان، دسترسی به ابزارهای ذخیره سازی، امنیت، کیفیت سرویس هایی مثل رزرو منابع و تلاش برای حداقل کردن هزینه را به عهده دارد.
- مدیر منبع محلی[۲]: مدیران محلی منابع مسئولیت مدیریت و زمانبندی اجرا به صورت محلی را به عهده دارند. همچنین وظیفه ایجاد دسترسی به وسایل ذخیره سازی، پایگاههای داده و سایر منابع اطلاعاتی را به عهده دارند. مثالهایی ازمنابع محلی را می توان سیستم کلاستر MOSIX و سیستم صف بندی Condor [۵۹] را نام برد.

۲-۳-۳- مشخصه های سیستم مدیریت منابع گرید محاسباتی

وجود انواع گریدها با هدفهای متفاوت، باعث شده تا سیستمهای مدیریت منابع آنها نیز دارای اجزاء ذاتا متفاوتی باشند. بررسی اجزاء اصلی سیستمهای مدیریت منابع انواع گریدها در این فرصت مختصر امکان پذیر نیست، بنابراین در این بخش سعی می کنیم مشخصه های اصلی سیستمهای مدیریت منابع گریدهای ابرمحاسباتی توزیع شده را که زمینه اصلی این پایان نامه را تشکیل می دهد، بررسی کنیم.

منابع گرید محاسباتی پردازنده ها یا منابعی می باشند که توسط سیستم مدیریت منابع، مدیریت می شوند. یک منبع محلی در گرید معمولا یک چندپردازنده یا یک کلاستر می باشد، که به لحاظ جغرافیایی در محلی قرار گرفته اند و از طریق شبکه های پر سرعت به هم مرتبط می باشند و در همان سازمان هم مدیریت می شوند. این منابع محلی ممکن است از یکدیگر دور بوده و با استفاده از اتصالات نامنظم اینترنتی با هم در ارتباط قرار گیرند. تمام این منابع تشکیل محیط ابر محاسباتی، به صورت آنچه در شکل۲-۵ نشان داده شده است، می باشد.

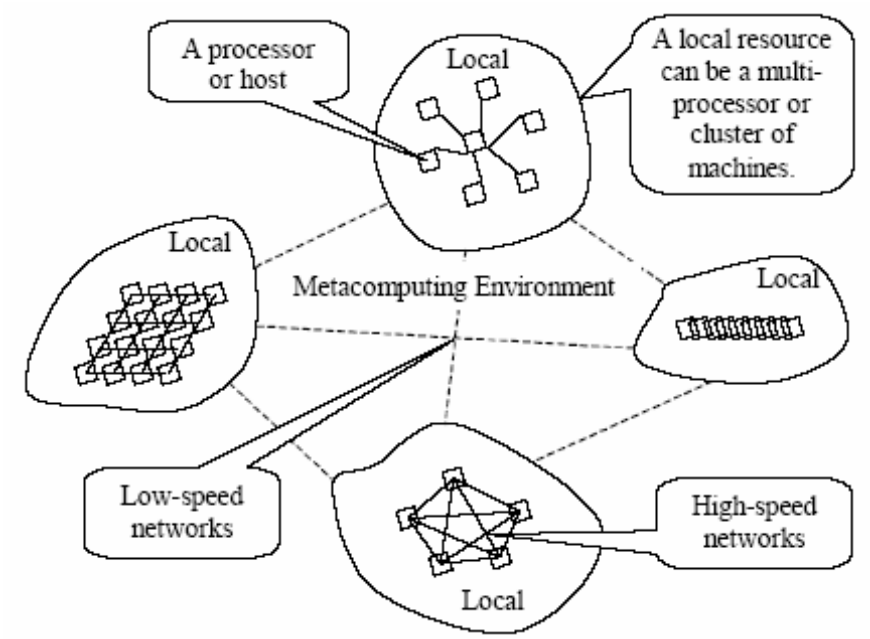
عملیات مربوط به مدیریت منابع گرید بایستی در هر دو سطح محلی و فرا محلی[۳] انجام شود. هر منبع محلی با مدیر منبع محلی خاص خود مدیریت می شود. مکانیزمی نیز در سطح فرامحلی برای هماهنگی رفتارهای مدیرهای محلی مختلف، برای نیل به کارایی بالا در سطح کل گرید مورد نیاز است.

مسائل اساسی مرتبط با مدیریت فرامحلی منابع عبارتند از ارائه و مدیریت داده، پروتکلهای ارتباطی، کشف منبع و پشتیبانی از کیفیت سرویس. مسائل اساسی مرتبط با مدیریت محلی منابع عبارتند از زمانبندی چند پردازنده ای، تخصیص منبع و نظارت بر کارکرد منابع. که این مباحث در زیر مورد بحث قرار گرفته اند.

الف) مدیریت داده: داده های اصلی مورد استفاده در سیستم مدیریت منابع داده هایی هستند که برای توصیف مشخصات و عملکرد یک منبع به کار برده می شوند. مسائل مرتبط با مدیریت داده عبارتند از ارائه و ذخیره سازی داده. یک منبع گرید می تواند به وسیله یک مدل منبع توصیف شود. مدل منبع مشخص می کند که چگونه منبع گرید توصیف گردد. دو رویکرد اساسی برای ارائه داده وجود دارند.

---

[۱] Grid Middleware
[۲] Domain Resource Manager
[۳] Meta level



شکل۲-۵. منابع مختلف موجود در گرید.[۳۳]

- مدل مبتنی بر شما: در این روش داده های توصیف کننده منبع، توسط یک زبان توصیفی بیان می شوند. این زبان ممکن است ثابت و یا توسعه پذیر باشد. مدل های از پیش تعریف شده که به صورت کلی مشخصه-مقدار هستند، جزء زبانهای توصیفی ثابت و زبان مشخصات منبع (RSL) که در گلوباس مورد استفاده قرار می گیرد و یا مدل ClassAd در کندر نمونه هایی از زبان توصیفی منابع به صورت توسعه پذیر هستند.
- مدل شیء گرا: در این مدل عملیات روی منابع به صورت جزئی از مدل منبع تعریف می شود. در لژیون از این رویکرد برای توصیف منابع استفاده شده است.

اطلاعات مربوط به منابع باید به صورت مناسبی ذخیره شوند. نحوه ذخیره سازی و دسترسی به این اطلاعات بر روی کارایی کلی سیستم مدیریت منابع تاثیر دارد. دو رویکرد اساسی برای ذخیره سازی این اطلاعات عبارتند از:

- فهرستهای شبکه ای. که معمولا بر اساس استانداردهای IETF مثل LDAP [۸۸] و SNMP [۳٤] هستند و یا از پایگاه داده های توزیع شده استفاده می کنند. MDS که در گلوباس استفاده می شود نیز از همین رویکرد استفاده کرده و در آن از استاندارد LDAP استفاده می شود.
- اشیاء توزیع شده. در این رویکرد عملیات و خصوصیات منبع در قالب کلی شیء تعریف می‌شوند. لژیون از مدل ذخیره سازی شیئی داده استفاده می کند.

از طرف دیگر برنامه های ارسالی کاربران نیز بایستی دارای مدل باشند که در آن مدل، اطلاعات نیازهای برنامه کاربر در مورد نحوه اجرا بیان شده است. اکثر مسائل فوق قابل تعمیم برای مدل برنامه نیز می باشد.

ب) پروتکلهای ارتباطی: ارتباطات، یک مساله حیاتی برای ایجاد سیستمهای نرم افزاری توزیع شده است. در محیط گرید نیز، مدیران محلی برای انجام مدیریت فرامحلی احتیاج به ارتباط با هم دارند. پروتکلهای ارتباطی به عنوان اساس اینگونه ارتباطات مورد نیازند تا قسمتهای مختلف بتوانند همدیگر را درک کنند.

ارتباطات می تواند توسط پروتکلهای اینترنت، از قبیل TCP/IP، FTP و HTTP پیاده سازی و به کار برده شوند.

ج) تبلیغ و کشف سرویس: عملیات اصلی یک سیستم مدیریت منابع گرید ارائه راهکاری برای پیدا کردن منابع درون گرید و بکارگیری آنها توسط برنامه های متقاضی است. عملیات کشف و تبلیغ سرویس در واقع مکمل هم هستند. فرآیند کشف توسط برنامه گرید (برنامه کاربر) راه اندازی می شود و برای پیدا کردن منبع مناسب و بکارگیری آن در درون گرید تلاش می کند. عملیات تبلیغ توسط منبع راه اندازی می شود و در آن، منبع سعی در یافتن پردازشی مناسب برای اجرا بر روی خود می کند. هزینه مربوط به تطبیق منابع و برنامه های متقاضی آنها، کارایی سیستم را مشخص می کند. دو رویکرد اساسی برای تبلیغ و کشف منابع در سیستمهای مدیریت منابع وجود دارد: روشهای مبتنی بر پرس و جو و روشهای مبتنی بر عامل.

- روش مبتنی بر پرس و جو: در روشهای فهرست شبکه ای مثل Globus MDS، پرس و جوهای پارامتری شده از طریق شبکه به نزدیکترین فهرست ارسال می شوند و سپس از یک موتور پرس و جو برای اجرای پرس و جو بر روی محتویات پایگاه داده استفاده می شود. روشهای مبتنی بر پرس و جو بسته به اینکه پرس و جو بر روی پایگاه داده توزیع شده و یا مرکزی اجرا می شود قابل دسته بندی اند. لژیون از کشف منبع به صورت پرس و جو بر روی پایگاه داده های توزیع شده عمل می کند حال آنکه روشهای متمرکز در پروژه هایی همچون کندر، نت سلو و نینف مورد استفاده قرار گرفته اند.
- روشهای مبتنی بر عامل: رویکردهای عامل گرا تکه کدهایی را از طریق ماشینهای درون گرید ارسال می کنند که این تکه کدها به صورت محلی بر روی هر ماشین تفسیر می شوند. عاملها همچنین می توانند به سایر عاملها پاسخ دهند و اطلاعات خود را در سطح شبکه توزیع کنند. بنابراین آنها تمام قابلیتهای روش مبتنی بر پرس و جو را دارا می باشند. در حال حاضر رویکردهای عاملگرا در پروژه های کشف سرویس همچون ۲k [۱۸] و Bond [۵۰] استفاده شده اند.

تفاوت عمده بین رویکرد عاملگرا و روشهای مبتنی بر پرس و جو در این است که سیستمهای عاملگرا به عامل اجازه کنترل و تصمیم گیری بر روی پرس و جو را بر اساس دانش داخلی عامل می دهد در حالیکه روشهای مبتنی بر پرس و جو بر اساس یک تابع (شرایط) ثابت عمل می کنند. روشهای عاملگرا به صورت ذاتی توزیع شده اند.

د) پشتیبانی از کیفیت سرویس (QoS)[۱] : در سیستم‌های گرید، کیفیت سرویس تنها به پهنای باند شبکه محدود نمی‌شود، بلکه تواناییهای پردازش و ذخیره سازی گره ها را نیز در بر می‌گیرد، بنابراین تمرکز، بر میزان کیفیت خدماتی است که گرید می‌تواند به صورت انتها-انتها ارائه کند. اگر وظیفه‌ای نیازمندیهای خاصی در زمینه کیفیت سرویس را بیان کند باید SLA مربوطه برای اجرای تعهدات مد نظر قرار گیرد. رزرواسیون منابع یکی از روشهای تضمین کیفیت سرویس می‌باشد. Globus تا حد زیادی رزرواسیون منابع را پشتیبانی می‌کند. گریدی که قابلیت تعریف کیفیت سرویس را در هنگام ارسال وظیفه فراهم می‌کند ولی امکان رزرواسیون منابع را ندارد، تنها راه حل ناقصی را برای تضمین کیفیت سرویس فراهم کرده است. بسیاری از گریدهای موجود تنها از کیفیت سرویس جزئی پشتیبانی می‌کنند، زیرا بسیاری از سیستم عاملها از تضمین دقیق اجرای وظایف پشتیبانی نمی‌کنند.

دو بخش عمده در تضمین کیفیت خدمات وجود دارد: الف) کنترل ارسال ب)نظارت

کنترل ارسال اطمینان حاصل می‌کند که آیا کیفیت مورد درخواست قابل ارائه است و از طرف دیگر آیا درخواست کننده صلاحیت دریافت چنین سرویسی را دارد یا نه و عدم تخطی وظیفه از سطح سرویس مورد توافق را کنترل می‌کند.

---

[۱] Quality Of Service

انواع خدمات کیفیت سرویس عبارتند از: الف)بدون کیفیت سرویس ب) نرم ج)سخت. سیستمهای مدیریت منابعی که امکان تعریف کیفیت سرویس مورد نظر را به وظایف می‌دهند ولی امکان نظارت را ندارند، تنها تضمین کیفیت خدمات نرم را فراهم کرده اند. اکثر سیستمهای فعلی از جمله گلوباس و نت سلو این سطح از کیفیت سرویس را ارائه می دهند. کیفیت سرویس سخت زمانی فراهم می‌شود که تمام گره های موجود در گرید بتوانند بر کیفیت سرویس توافق شده، نظارت کنند. Nimrod/G خدمات کیفیت سرویس سخت ارائه می دهد.

ه) زمانبندی منابع: زمانبندی بر روی یک منبع محلی در گرید یک مساله "چند برنامه بر روی چند پردازنده" است. برنامه ها در زمانهای متفاوت و با نیازهای متفاوت می رسند. و زمانبند محلی مسئول مشخص کردن زمان اجرای برنامه ها است. دو نوع سیاست زمانبندی و معیارهای مربوط به آنها وجود دارد:

- سیاست منبع گرا: که به دنبال حداکثر کردن بهره وری منابع است.
- سیاست برنامه گرا: که در آن هدف برآوردن نیازهای برنامه کاربر است.

این دو جنبه زمانبندی به هم مرتبط بوده و بعضا ممکن است متناقض نیز باشند. بنابراین بایستی راه حلی برای ارضاء همزمان آنها پیدا کرد. زمانبندی مجدد نیز قسمتی از مساله زمانبندی است که مشخص می کند چه موقع زمانبندی فعلی دوباره انجام می شود. در این رابطه نیز دو رویکرد وجود دارد:

- رویکردهای زمانبندی مجدد دوره ای و دسته ای، تقاضاهای بکارگیری منابع و رخدادهای سیستم را دسته بندی کرده و آنها را در فواصل زمانی انجام می دهند. این فاصله زمانی ممکن است دوره ای باشد و یا با رویدادهای سیستم فعال شوند. نکته اصلی آن است که زمانبندی مجدد به ازاء هر تقاضا یا رویدادی انجام نمی شود.
- رویکرد رویدادگرا، که در آن به محض ورود یک تقاضای جدید و یا بروز رویداد سیستمی، زمانبندی مجدد انجام می شود.

و) تخصیص منبع و نظارت: پس از اینکه برنامه ها بر روی یک منبع گرید زمانبندی شدند، قسمت مربوط به تخصیص منبع موظف به اجرای برنامه و برگرداندن نتایج است. قسمت مدیریت محلی منابع بایستی با ابزارهایی همچون MPI وPVM تجهیز شوند. وقتی برنامه ای شروع به اجرا بر روی یک منبع می کند، بایستی در حین اجرا بر وضعیت منبع نظارت شود. نتایج حاصل از این نظارت در سطح مدیریت محلی برای زمانبندی مجدد و در سطح فرامحلی برای کشف سرویس بکار می رود.

۲-٤- کاربرد عاملها در سیستمهای مدیریت منابع

زیر ساخت نرم افزاری گرید، سیستمی باز و پیچیده است. راهکارهای چندعامله[1] یکی از راههای فائق آمدن به مشکلات مطرح در توسعه گرید است. سرویس به عنوان مهمترین مفهوم در سیستمهای توزیع شده پذیرفته شده است، و بنابراین کشف سرویس به عنوان جزء ضروری بسیاری از زیرساختهای گرید می باشد. در این بخش به اجمال تحقیقات انجام گرفته برروی کشف سرویس در محیطهای چند عامله را بررسی می کنیم.

[1] Multi-Agent

۲-٤-۱- سیستمهای چند عامله

با توجه به گستردگی تحقیقات صورت گرفته و کاربرد سیستمهای چند عامله، بررسی آنها به طور مختصر امر دشواری است. با این حال در زیر سعی می شود یک تقسیم بندی کلی از مباحث مطرح در این سیستمها ارائه شود.

هر عامل موجودیتی خود مختار است که دارای داده ها، منابع و محیط خاص خود می باشد. مشخصه اصلی یک عامل، استفاده از داده های داخلی خود در سطح دانش است. بر اساس این دانش، ممکن است عاملها بخواهند با هم ارتباط برقرار کنند. KQML می تواند به عنوان زبان ارتباطی عاملها[۱] مورد استفاده قرار گیرد.

حال به بیان جزئی تر هر یک از این مفاهیم می پردازیم.

ارائه دانش یک عامل، نگاشتی بین وضعیت محیط خارجی و سیستم استنتاج داخلی عامل می باشد. ابزارهای عمومی ارائه دانش، شامل یادگیری به صورت کارا، حل مساله به صورت کارا و استنتاج معقول عامل می باشد.

ارتباط عاملها، همانطور که گفته شد این ارتباط ممکن است از طریق یک زبان استاندارد و یا از طریق هر توافق دیگری بین عاملها صورت گیرد.

مذاکره عاملها، عبارتست از فرایندی که بر اساس آن دو عامل بر روی مساله خاصی به توافق می رسند.

هماهنگی عاملها، که یکی از مهمترین قسمتهای سیستم چند عاملی است و عبارتست از مدیریت تراکنشها و وابستگیهایی بین فعالیتهای مختلف عاملها [۸٤].

مسائل جزئی تر در مورد تـئوری، کاربردها، روشها و ابزارهای سیستمهای چندعامله در مرجع [۸۹] بحث شده اند.

۲-٤-۲-کاربرد عاملها

همانطور که در بخش قبل اشاره شد، امروزه سیستمهای چند عامله کاربردهای متنوعی در علوم کامپیوتر و نرم افزار دارند. از آن جمله می توان به روشهای مهندسی نرم افزار عاملگرا[۲] [۷۰،۵۷] که مکمل گامهای قبلی همچون مدلهای ساخت یافته و مدلهای شیی گرا هستند و به ویژه در زمینه طراحی نرم افزارهای توزیع شده کاربرد دارند، اشاره کرد.

عاملها دارای هوشمندی، رفتار اجتماعی و خودمختاری هستند. همه این خواص می توانند در محیط گرید کاربرد داشته باشند. به عنوان مثال یک زمانبند منبع که موجودیت مهمی در سیستم مدیریت منابع گرید است به علت بزرگی فضای جستجوی خود ممکن است از روشهای هوش مصنوعی برای حل مسائل خود استفاده کند.

هدف کلی محیطهای چند عاملی، ارائه یک لایه تجرید سطح بالا برای مدیریت منابع گرید است. به عنوان مثال عاملها در چندین پروژه گرید از قبیل Apples, DPSS و NetSolve به کار رفته اند. اخیراً نیز یک پروژه مبتنی بر عامل گرید به نام "Agent Grid" ایجاد شده است [٦٥]. در این کار تحقیقاتی، از عاملها برای مجتمع سازی سرویسها و منابع و ایجاد محیطهای حل مساله چند وجهی استفاده شده است. در متدلوژی A٤ که در بستر مدیریتی ARMS به کار رفته است نیز از تجرید عاملها برای ترکیب مفهوم سرویس و منبع استفاده شده است. در این سیستم به جای استفاده از مجموعه ای از عاملها که هر یک تخصص و کاربرد خاص خود را دارند، از سلسله مراتبی از عاملهای غیر متحرک که در آن واحد نقش سرویس دهنده، سرویس گیرنده و میانجی را اجرا کنند، استفاده شده است.

به طور کلی می توان گفت که با توجه به شرایط محیط گرید و ویژگیهایی که سیستمهای چندعاملی در اختیار قرار می دهند، استفاده از عاملها برای ایجاد سیستم مدیریت منابع گرید، انتخاب خوبی است و می تواند بسیاری از نقائص

---

[۱] ACL(Agent Communication Language)

[۲] AOSE: Agent Oriented Software Engineering

سیستمهای قبلی را مرتفع سازد. با این نتیجه گیری در قسمت بعد به بررسی خواص چنین سیستمهایی پرداخته و در همان راستا مدل مدیریت منابع چند عاملی به نام ARMS [۳۳] را معرفی می کنیم و ویژگیها و نقائص آنرا مورد بحث قرار می دهیم. این سیستم از آن جهت که به عنوان قالب کاری ما در طی این پایان نامه می باشد و قصد بهینه سازی قسمتهایی از آن را داریم برای ما مهم است.

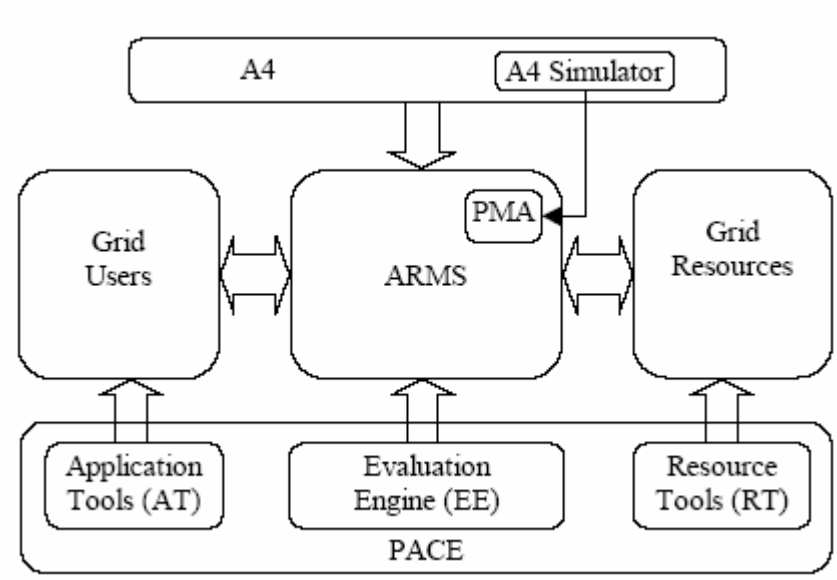
۲-۵- روش مدیریت منبع مبتنی بر عامل (ARMS)

از بخشهای قبل می توان نتیجه گرفت که مدیریت منابع در گرید های محاسباتی تا حد زیادی متکی به توانایی در پیش بینی صحیح و دقیق کارایی برنامه های آن است. همچنین اشاره شد که در چنین محیطهایی قیاس پذیری و تطبیق پذیری دو مساله مهم و حیاتی به شمار می روند. در این بخش یک راهکار مدیریت منابع مبتنی بر عامل برای گریدهای محاسباتی به نام ARMS [۳۳] بررسی می شود که در آن سعی شده است هدفهای فوق برآورده شوند.

ARMS در سال ۲۰۰۱ در پایان نامه دکترای فردی به نام J.Cao ارائه شد. در این سیستم مدیریت منابع از سلسله مراتبی از عاملهای همگون [۳۵] که مجهز به ابزاری برای پیشگویی کارایی، به نام PACE [٤١] می باشند، استفاده شده است. در بخشهای بعدی این سیستم مدیریت منابع را به طور کامل بررسی می کنیم.

۲-۵-۱- پیش زمینه ARMS

ARMS به طور کلی پلی برای زمانبندی برنامه ها و بکارگیری منابع موجود گرید بین کاربران و منابع گرید برقرار می کند. این سیستم از قسمتهای مختلفی تشکیل شده است. رابطه بین اجزاء مختلف ARMS و سایر مفاهیم مطرح در آن در شکل ۲-٦ نشان داده شده است.

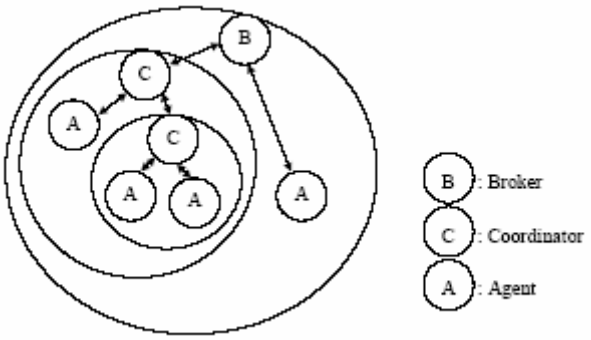


شکل ۲-٦. ساختار کلی مدل مدیریت منابع ARMS [۳۳].

از PACE برای ارائه داده های کمی در رابطه با کارایی برنامه های پیچیده ی در حال اجرا بر روی منابع محلی استفاده می شود. دراینصورت برنامه ای که بخواهد توسط کاربر اجرا شود بایستی دارای یک مدل برنامه[۱] باشد. این مدل با استفاده از یکی از قسمتهای PACE به نام "ابزار برنامه"(AT) فراهم می گردد. همچنین ابزاری در هر منبع نیز باید وجود داشته باشد که مدل منبع[۲] (اطلاعات کارایی) خاص آن منبع را ارائه نماید. در PACE به این قسمت "ابزار منبع" (RT) گفته می شود. قسمت دیگری از PACE که در درون عاملها به کار رفته و عملیات ارزیابی کارآیی را انجام می دهد، "موتور ارزیابی[۳]" (EE) نامیده می شود. این قسمت که وظیفه اصلی PACE را اجرا می کند، اطلاعات منبع(RT) و اطلاعات برنامه(AT) را به عنوان ورودی دریافت کرده و میزان کارایی اجرای یک برنامه خاص بر روی یک منبع خاص را برمی گرداند. به عنوان مثال می تواند میزان زمان لازم برای اجرای یک برنامه بر روی یک منبع را به عنوان خروجی بدهد. کارایی PACE در چندین حالت و با استفاده از برنامه ها و منابع مختلف مورد آزمایش قرار گرفته و صحت نتایج آن تایید شده است [۳۶]. با توجه به صحت این نتایج، می توان از نتایج PACE در مدیریت منابع گرید نیز استفاده کرد.

در سطح بالاتر قسمتی که وظیفه انجام عملیات مدیریت منابع در سطح کل گرید را بعهده دارد باید وجود داشته باشد. در ARMS از متدولوژی A٤[٤] برای انجام عملیات مدیریت منابع در سطح عاملها استفاده می شود[۳۷].

A٤ باعث می شود عاملها در یک ساختار سلسله مراتبی(درختی) نسبت به یکدیگر قرار بگیرند. این مساله تا حد زیادی مشکل قیاس پذیری در گرید را مرتفع می سازد. همه عاملها در این سلسله مراتب دارای یکسری عملیات یکسان هستند. در شکل۲-۷ از عبارات مختلفی برای متمایز ساختن عاملهای سطوح مختلف استفاده شده است. واسطه[۵] عاملی است که در راس درخت عاملها قرار دارد و طبیعتاً اطلاعات کل سیستم را در خود دارد. هماهنگ کننده[٦] ریشه هر زیر درخت است و یک گره برگ نیز یک عامل عادی است.



شکل۲-۷.سلسله مراتب عاملها در مدل مدیریت [۳۳].

علی رغم مکانهای مختلف و اسامی مختلف عاملها در این سلسله مراتب، همه عاملها عملکرد یکسانی دارند. این سلسله مراتب از عاملهای همگون باعث به وجود آمدن یک تجرید سطح بالا از سیستم توزیع شده می شود. این ساختار درختی همچنین باعث امکان پویایی در سیستم مدیریت منابع می شود. عاملهای جدید می توانند به درخت بپیوندند و عاملهای فعلی می توانند درخت را در هر زمانی ترک کنند.

---

[۱] Application Model
[۲] Resource Model
[۳] Evaluation Engine
[٤] Agile Architecture Autonomous Agent
[۵] Broker
[٦] Coordinator

در این درخت هر گره فقط از شرایط گره های همسایه خود باخبر است؛ در اینصورت هر عامل در یک گره پدر و چندین گره فرزند ثبت شده است. بنابراین در هنگام ترک سیستم، عامل کافیست به گره پدر اطلاع دهد و از گره های فرزند بخواهد در پدر جدید مجددا خود را ثبت کنند.

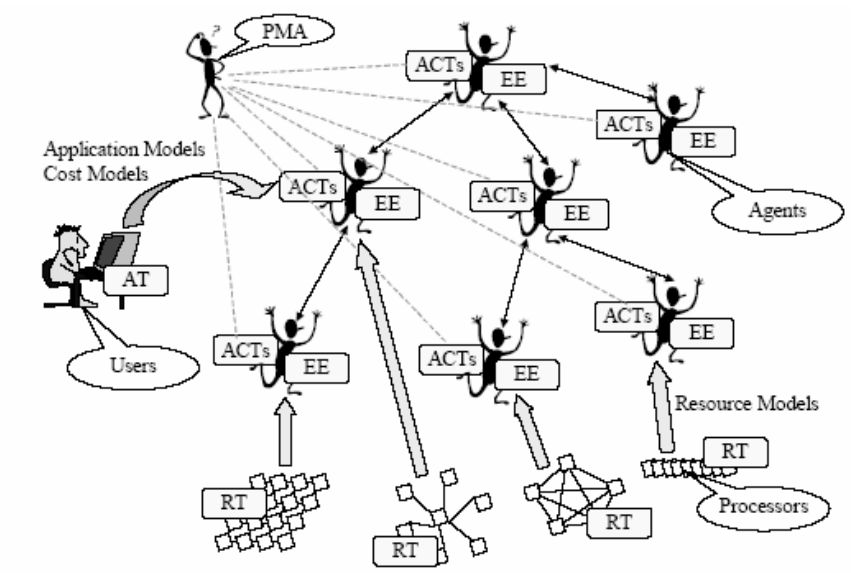
عاملها وظیفه انجام تبلیغ و کشف سرویس را نیز به عهده دارند. همه عاملها در سازمان ARMS ساختار یکسانی دارند، در عین حال رفتار این عاملهای همگون باید به گونه ای تنظیم شود که کارآیی مطلوبی را نتیجه دهند. برای نیل به این هدف در ARMS عامل خاصی به نام "عامل ناظر و مشاورکارایی[۱] (PMA) وجود دارد که قسمت اصلی آن، همان شبیه ساز A٤ است. وظیفه این عامل نظارت و تنظیم رفتار سایر عاملها، به گونه ای است که باعث افزایش کارایی کل سیستم شود.

در سیستم ARMS اولیه فقط پردازشهایی با خواص بلادرنگ قابل اجرا بودند. یعنی زمانبندهای داخلی ARMS قابلیت انعطاف کافی برای برخورد با انواع پردازشهای گوناگون را نداشتند. به عنوان مثال اگر پردازشی نیاز به محدودیت زمانی خاصی در اجرا نداشت (همانند پردازشهای دسته ای)، می بایست باز هم با یک محدودیت زمانی صوری تجهیز می شد. بعدها با اضافه کردن زمانبندی به نام COSY انعطاف پذیری بیشتری به ARMS در برخورد با پردازشهای گوناگون داده شد.

۲-۵-۲- معماری ARMS

همانطور که گفته شدARMS یک سیستم مدیریت منابع مبتنی بر عامل است. در شکل۲-۸، شکل کلی معماری این سیستم نشان داده شده است.

همان طور که بیان شد اجزای اصلی این معماری شامل کاربران و منابع گرید، عاملهای همگون و نوع خاصی عامل به نام PMA است. در زیر این موارد شرح داده می شوند.



شکل۲-۸ معماری و نحوه تعامل عاملها در ARMS [۳۳].

[۱] Performance & Monitor Advisor

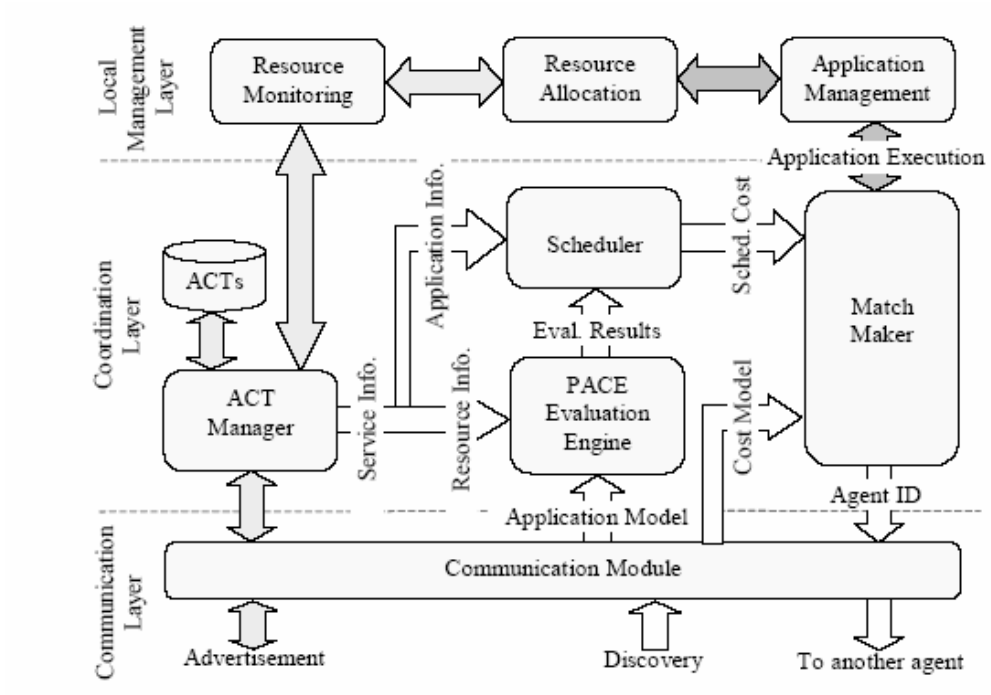
- کاربران گرید:برنامه نویسان عبارتند از کسانیکه از ابزارهای موجوداستفاده کرده وبرنامه هایی برپایه گرید می نویسند. این برنامه ها به ویژه آنهایی که با MPI و PVM ایجاد شده اند نیاز زیادی به وقت CPU و حافظه دارند. قبل از اجرای یک برنامه تحت گرید، ابتدا برنامه کاربر از ابزار AT برای بدست آوردن مدل کارایی لازم برای برنامه استفاده می کند. این کار به خاطر سادگی عملیات PACE می تواند توسط کاربران عادی نیز انجام شود. بنابراین هر تقاضایی که قصد اجرا دارد، یک مدل (حاصل از AT) نیز باید با خود داشته باشد. جزء دیگری که همراه درخواست کاربر است مدل هزینه است که تمام اطلاعات در مورد اجرای برنامه کاربر، مثل فرصت زمانی[۱] و... را در بر دارد.
- منابع گرید: که شامل هر منبع محاسباتی، از سیستمهای موازی بزرگ گرفته تا کلاسترها و حتی PC ها می باشد. در ARMS هر منبعی دارای مدل کارایی خاص خود است. این مدل با استفاده از "ابزار منبع" (RT) قابل حصول است. RT با استفاده از یکسری برنامه محک[۲] اطلاعات کارآیی منبع را بدست می آورد. این اطلاعات بعداً می تواند برای تبلیغ منبع مورد استفاده قرار گیرند.
- عاملهای ARMS : از اجزاء اصلی ARMS است. هر عاملی در ARMS نماینده یک یا چند منبع خاص، به صورت سطح بالاست. این عاملها در A٤ استفاده می شوند. هرعامل دارای جداولی به نام "جدول توانایی عامل[۳]" (ACT) است که در آن علاوه بر اطلاعات توانائیهای منابع خود عامل، اطلاعات سرویس سایر عاملهای همسایه نیز نگه داشته می شود. این اطلاعات شامل تمام اطلاعات کارآیی منبع است و برای محاسبه کارآیی مورد استفاده قرار می گیرند. موتور ارزیابی PACE نیز درون هر عامل گنجانده شده است. توانایی پیشبینی کارآیی موتور ارزیابی (EE) ، می تواند برای مدیریت منابع محلی در زمانبندی برنامه های موازی بکار رود. این قسمت همچنین در لایه هماهنگی عاملها مورد استفاده قرار می گیرد تا توانایی تضمین کیفیت سرویس را در کشف منابع ایجاد کند. همانطور که گفته شد، عاملهای ARMS همگون هستند. این بدان معناست که هر عامل در آن واحد می تواند به عنوان نماینده تقاضای کاربر(متقاضی منبع)، نماینده منبع (ارائه دهنده منبع) و یا واسطه[٤] بین این دو باشد.
- PMA : عاملها که در ARMS به صورت سلسله مراتبی قرار می گیرند، تشکیل درختی می دهند که در ریشه آن عاملی با ساختار متفاوت وجود دارد؛ این عامل همان عامل PMA است. همانطور که در شکل قبل نشان داده شد، این عامل با تمام عاملهای دیگر ارتباط برقرار نموده و با استفاده از شبیه ساز A٤ که در درون این عامل قرار دارد، کارآیی عمل کشف سرویس را بهبود می بخشد.

۲-۵-۳- ساختار عاملها

این ساختار به طور کامل در شکل۲-۹ نشان داده شده است.

---

[۱] Deadline
[۲] Benchmark
[۳] Agent Capability Table
[٤] Matchmaker



شکل۲-۹. ساختار درونی هر عامل در ARMS [۳۳].

همانطور که در شکل۲-۹ دیده می شود، عاملها از سه لایه ارتباط، هماهنگی و مدیریت محلی تشکیل شده اند. هر لایه شامل ماژولهای زیادی است که از تعامل آنها عملیات کشف و تبلیغ سرویسها انجام می شود.

لایه ارتباط به عنوان واسطه ای بین عامل و دنیای خارج عمل می کند. از طریق این دریچه ممکن است پیغامهای مربوط به کشف و تبلیغ سرویس دریافت شوند. این لایه محتوای پیغامها را تحلیل کرده و به قسمتهای متناظر در لایه هماهنگی ارسال می نماید. مثلاً یک پیغام تبلیغ مستقیماً به مدیر ACT در لایه هماهنگی تحویل داده می شود. همچنین این لایه وظیفه ارسال پیغام کشف و تبلیغ سرویس به سایر عاملها را نیز به عهده دارد.

در لایه هماهنگی چهار قسمت مختلف شامل مدیر ACT، موتور ارزیابی PACE ، زمانبند، و تطبیق دهنده قرار دارد که از تعامل این قسمتها نحوه برخورد با یک پیغام دریافت شده مشخص می شود. مثلاً جوابی که به یک پیغام اکتشاف داده می شود ممکن است اجرا بر روی منابع محلی و یا اعزام به سایر عامل های همسایه باشد.

در لایه مدیریت محلی منابع، وظیفه مدیریت برنامه، تخصیص منبع و نظارت بر منبع انجام می گیرد. دستور اجرای برنامه از لایه هماهنگی به لایه مدیریت محلی می رسد. در این دستور اطلاعات زمان بندی، مثل زمان شروع اجرا و شماره پردازنده های تخصیص یافته قرار دارند. این برنامه ها در قسمت مدیریت برنامه ها[۱] صف بندی می شوند تا بر روی منابع محلی اجرا شوند. وقتی زمان شروع یک برنامه می رسد، برنامه به قسمت تخصیص منبع فرستاده می شود، در این قسمت تطبیق دهنده هایی با انواع محیطها مثل MPI و PVM قرار دارند و اجرای برنامه به طور واقعی شروع می شود. یکی از قسمتهای مهم لایه مدیریت محلی منابع، قسمت نظارت بر منابع است. این لایه به صورت دوره ای، برنامه های محک PACE را اجرا کرده و اطلاعات کارایی منابع (RT) را بدست می آورد. این قسمت همچنین وظیفه دارد با مدیریت برنامه ها و قسمت تخصیص منابع برای اطلاعات منابع و برنامه های باقیمانده ارتباط برقرار کند. به اینصورت اطلاعات ACT ها شکل می گیرد.

[۱] Application Management

۲-۵-٤- نحوه تبلیغ و اکتشاف سرویس

عمل تبلیغ سرویس فرآیندی است که طی آن یک منبع به دنبال برنامه ای برای اجرا بر روی خود می گردد. همانطور که گفته شد یک عامل ممکن است مدیریت چندین منبع را به عهده داشته باشد. لایه مدیریت محلی وظیفه جمع آوری اطلاعات سرویس ها و انتقال آن به لایه هماهنگی را به عهده دارد. در این لایه داده ها در جداول ACT ذخیره می شود. عامل ممکن است اطلاعاتی را نیز از محیط خود (سایر عاملها) دریافت کند و از آنها به عنوان دانش داخلی خود استفاده نماید. جداول ACT به دسته های T-ACT که اطلاعات منابع همان عامل را نگه می دارد و L-ACT که اطلاعات منابع عاملهای زیرین در سلسله مراتب عاملها را نگه می دارد و جداول G-ACT که اطلاعات سرویس طبقه بالاتر در سلسله مراتب عاملها را نگه می دارد تقسیم بندی می شوند. درC-ACT نیز نتایج جستجوهای اخیر عامل وجود دارد که ممکن است در جستجوهای آینده مفید واقع شوند. نحوه نگهداری و بروزسازی جداول ACT به روشهای مختلفی ممکن است انجام شود. در روش فراخواندن داده[۱] با تغییر در یک قسمت، تغییر به بقیه قسمتها اعلام نمی شود. بلکه عاملها با مراجعه به عامل تغییر یافته، می توانند تغییر را بر روی جداول خود بروز کنند. در روش ارسال داده[۲] بلافاصله پس از تغییر، این تغییر به تمام اطرافیان اطلاع داده می شود. هر کدام از این روشها می تواند به صورت دوره ای یا رویداد گرا پیاده سازی شود. در ARMS مدل "ارسال داده به صورت دوره ای[۳] " استفاده شده است.

فرآیند کشف سرویس با رسیدن یک درخواست به عامل آغاز می گردد. یک درخواست شامل چند قسمت می باشد:

- اطلاعات درخواست[٤] : شامل جزئیات سرویسهایی که کاربر علاقمند است پیدا نماید. این اطلاعات ممکن است با اطلاعات سرویسها در ACT ها ترکیب شود.
- نیازمندیها[٥] : حاوی اطلاعات کارآیی مورد نیاز از طرف کاربر است که برای تطبیق بین عاملها استفاده می شود تا مشخص کند که آیا یک منبع می تواند سرویس مورد نیاز را ارائه دهد یا خیر.
- موارد اختیاری[٦]: مواردی که وجود آنها الزامی نیست. مثلاً کاربر ممکن است زمان و حوزه فرآیند اکتشاف منبع را محدود کند.

یک عامل در برابر درخواستی که از او می شود به روشهای مختلفی می تواند پاسخ دهد. در صورت مثبت بودن پاسخ، سرویس مورد نظر پیدا شده است. در صورت پیدا نشدن منبع، عامل می تواند مکان جدیدی را برای جستجو پیشنهاد دهد. در فرآیند تصمیم گیری از اطلاعات جداول ACT استفاده می شود. در واقع فرآیند اکتشاف سرویس در عامل همان فرآیند جستجـــو در ACT ها می باشد. یک عامــل برای این منظور به ترتیب جـداول L-ACT، C-ACT ، T-ACT و در نهایت G-ACT را جستجو می کند. جواب مثبت وقتی داده می شود که نتایج درT-ACT یافت شود. اما در غیـر اینصورت از C-ACT استفاده می شود که در آن نتایج جستجوهای قبلی وجود دارد. در صورت پیدا شدن سرویس درC-ACT، تقاضا به عامل متناظر سرویس فرستاده می شود و در غیر اینصورت به جدول L-ACT مراجعه می شود که در آن اطلاعات سرویس حوزه محلی (فرزندان عامل) قرار دارد. در صورت یافت نشدن سرویس در نهایت به سراغ G-ACT ها که حوزه اطلاعاتی بیشتری را در بر می گیرد می رویم. اگر پس از جستجو در این جداول سرویس مناسب پیدا نشد می توان تقاضا را به عامل سطح بالاتر فرستاد که با احتمال بیشتری سرویس مناسب

---

[۱] Data Pull
[۲] Data Push
[۳] Periodic Data Push
[٤] Request Information
[٥] Requirements
[٦] Options

یافت می شود.در نهایت پس از جستجوی ACT های تمام طبقه ها ممکن است سرویس یافت نشود و فرآیند کشف سرویس شکست بخورد و این زمانی است که تقاضا به ریشه سلسله مراتب عاملها رسیده است.

۲-٦- جمع بندی

محاسبات گرید، همان محاسبات توزیع شده است که پا به مرحله تکامل نهاده و هدف آن ارائه کامپیوتر مجازی بزرگ و خود مدیریت شونده است. سیستم مدیریت منابع، مؤلفه اصلی سیستمهای گرید می‌باشد. در گرید، منابع متولیان مختلفی دارند، بنابراین سیستم مدیریتی مورد استفاده باید مورد توافق تمامی صاحبان منابع باشد. به علت بدون مرز بودن و گستردگی گرید، استفاده از روشهای مدیریتی مرکزی، کارآیی و قیاس پذیری سیستم را به شدت مورد تهدید قرار داده و سیستم را آسیب پذیر می نماید. علی رغم شباهتهای زیاد سیستمهای توزیع شده و گرید، به خاطر شرایط محیطی کاملا متفاوت با سیستمهای توزیع شده، تفاوتهای عمده ای در سیستمهای مدیریت منابع آنها وجود دارد. مسائلی همچون مدیریت حجم زیاد و نا مشخصی از داده های مدیریتی، احتیاج به پروتکلهای ارتباطی شبکه ای و بین شبکه ای کارا، تغییرات و عدم پایداری منابع، کشف و تبلیغ بهینه منابع و زمانبندی منابع در هر دو سطح محلی فرامحلی از جمله این تفاوتها هستند که سیستم مدیریت منابع باید بر آنها فائق آید. محیطهای چند عامله با شرایط چنین محیطی همخوانی دارند. نمونه ای از سیستمهای مدیریت منابع که مبتنی بر عاملها و محیطهای چند عاملی می باشد ARMS است. در این سیستم، عاملها در یک ساختار درختی قرار گرفته و هر عامل به ابزاری برای پیشگویی کارایی به نام PACE مجهز است. عملیات مدیریت منابع در ARMS با همکاری عاملها انجام می شود. در این متد مدیریتی دو مشکل اساسی گرید یعنی قیاس پذیری و تطبیق پذیری تا حد زیادی برطرف شده است.

با این حال ARMS هنوز در ابتدای راهی طولانی است و قسمتهای مختلف آن احتیاج به بهینه سازی دارد. این کاستیها به ویژه در مباحث زمانبندی منابع و استفاده بهینه و متوازن از تمام منابع بیشتر به چشم می آید.

فصل ۳

موازنه بار در گریدهای محاسباتی

۳-۱- مقدمه

برای پاسخگویی به نیاز روزافزون برنامه های علمی، همانند برنامه های مربوط به علم فیزیک، ستاره شناسی، علوم زیستی و سایر کاربردها مفهوم "محاسبات گریدی" در اواسط دهه ۱۹۹۰ پا به عرصه نهاد و هدف اصلی آن سرجمع کردن نیروی محاسباتی بیکار در اینترنت و تبدیل آن به یک نیروی محاسباتی عظیم برای همه کاربران در سراسر دنیا بود[۱۹،۲۰].

یکی از مسائل مهم در چنین محیطی مدیریت منابع می باشد. مدیریت منابع یکی از اجزاء مهم و زیر ساخت محیط گرید است که در فصل قبل در مورد آن به تفصیل صحبت شد. هدف کلی مدیریت منابع، زمانبندی کارای برنامه هایی است که برای اجرا نیاز به استفاده از منابع موجود در محیط گرید را دارند. یکی از کارهای اصلی یک مدیر منابع گرید ارائه مکانیزمی برای کشف منابع درون محیط گرید و به کارگیری آنها توسط برنامه هاست. این عملیات از طریق فرآیند کشف و تبلیغ انجام می شوند. عملیات کشف توسط برنامه انجام می شود تا منبع مناسب برای خود را درون گرید پیدا کند. عملیات تبلیغ بالعکس توسط منبع صورت می گیرد و طی آن منبع سعی می کند برنامه مناسبی را برای اجرا بر روی خود بیابد. تطبیق گر، یک میان افزار است که سعی در تطبیق برنامه ها و منابع دارد. تطبیق گر ممکن است به صورت مرکزی یا توزیع شده پیاده سازی شود. از آنجایی که گرید ذاتا یک محیط پویا و بدون مرز می باشد، در کل روشهای توزیع شده نتایج بهتری را نتیجه می دهند و علاوه براین قیاس پذیرترند[٦٦،۳۳]. یک تطبیق گر ایده ال باید با استفاده از روشهای موازنه بار تقاضا ها را به صورت یکنواخت در سطح منابع گرید پخش کند. بنابراین باید مجهز به روالهای خاصی جهت **موازنه بار** باشد.

در یک تعریف کلی، هدف الگوریتمهای موازنه بار، توزیع یکسان بار بر روی منابع و حداکثر کردن کارایی آنها و در ضمن کم کردن زمان اجرای کلی کارهاست [٤]. در تعریف دیگری الگوریتم موازنه بار ایده ال، الگوریتمی است که در نهایت باعث می شود تمام گره ها در یک زمان کارهای خود را تمام کنند [۷].

مسئله موازنه بار برای محیط گرید که در آن یکی از مهمترین مسائل، توزیع عادلانه بار بر روی منابع می باشد یک ضرورت اساسی محسوب می شود.

در این فصل علاوه بر آشنایی با موازنه بار، ضرورت انجام آن در محیط گرید، عملیات لازم برای انجام آن و انواع روشهای موجود برای آن در محیط گریدهای محاسباتی و پیشینیان آنها، یعنی سیستمهای توزیع شده مورد بررسی قرار گرفته و در مورد مزایا و معایب هر یک به تفصیل بحث خواهد شد.

۳-۲- موازنه بار

چون گرید تعداد بسیار زیادی از منابع را که در فضای بسیار گسترده ای به لحاظ جغرافیایی قرار گرفته اند، به هم متصل می کند و تقاضاهای کاربران نیز به صورت توزیع شده به گرید وارد می شوند، یک مساله مهم، چگونگی توزیع تقاضاهای ارسال شده کاربران به واحدهای محاسباتی به صورت یکنواخت است. با حل این مشکل می توان با بهره وری بالایی از منابع محاسباتی استفاده کرد [٤۲]. دانشمندان به این نتیجه رسیدند که روشهای زیادی که برای توزیع بار[۱] ارائه شده اند به دو دسته بسیار کلی قابل تقسیم اند:

الف) الگوریتمهای موازنه بار[۲]، که هدفشان یکسان سازی بار پردازشی بین تمام گره های گرید می باشد.

[۱] Load Distribution
[۲] Load Balancing

ب) الگوریتمهای تقسیم بار[۱]، که هدف آنها ممانعت از بیکار ماندن بعضی از گره ها در شرایطی است که در سایر گره ها تقاضاها صف کشیده اند. به عبارت دیگر در این روشها الزاما به دنبال یکسان سازی کامل بار نیستند. مسلما در این روشها بار بسیار کمتری برای جابجایی پردازشها به زیر ساخت شبکه ای وارد می شود[٦٦].
البته این دسته بندی بسیار کلی بوده و اصطلاحات فوق بعضاً به جای هم نیز به کار می روند.

۳-۲-۱- ویژگیهای الگوریتم موازنه بار

ویژگیهای یک راهکار موازنه بار مطلوب عبارتند از: قیاس پذیری[۲]،تطبیق پذیری[۳]، پایداری[٤]، شفافیت از دید برنامه کاربر[٥]، دارای قابلیت تحمل در برابر خطا[٦] و حداقل بودن هزینه سربار تحمیلی به سیستم [١٤].
قیاس پذیری عبارتست از وابسته نبودن الگوریتم به اندازه وتوپولوژی فیزیکی سیستم. البته حتی اگر روش مورد استفاده کاملا هم قیاس پذیر باشد، اندازه سیستم حداقل بر روی میزان تاخیر در شبکه تاثیر می گذارد. با توجه به حساس بودن الگوریتمهای موازنه بار به داشتن اطلاعات صحیح و بروز از کل یا جزئی از سیستم، افزایش میزان تاخیر در شبکه به معنی از بین رفتن اطلاعات بروز در مورد گره های سیستم (کهنگی اطلاعات) و در نتیجه عدم کارکرد مطلوب الگوریتم موازنه بار می شود. تا کنون مطالعات زیادی بر روی مساله تاخیر و در نتیجه قابل اتکا نبودن اطلاعات موجود در گره ها انجام شده و راه حلهای چندی نیز ارائه شده است که این راه حلها تا حدی بر مشکل کهنگی اطلاعات غلبه می کنند [٧٧،٥٥،٦٠]. به عنوان مثال در [٥٥] پیشنهاد شده است که گیرنده به جای دریافت اطلاعات بار سایرین، با توجه به اطلاعات قبلی خود از آن گره میزان بار فعلی آن را حدس بزند: دراینصورت احتیاج خیلی کمی به تبادل بار داریم. متاسفانه علی رغم اهمیت مساله تاخیر در ارائه راه حل موازنه بار مناسب، در اکثر روشهای فعلی از این مساله صرف نظر شده است. باید در نظر داشت که صرفنظر از این مساله مهم، راه حل ارائه شده را تا حد زیادی از واقعیت به دور می سازد.
پایداری الگوریتم، پیش شرطی برای قیاس پذیری محسوب می شود و عبارتست از توانایی الگوریتم در اجتناب از اتخاذ تصمیمات ضعیف وبعضاً نادرست. مسلما علاوه بر خود الگوریتم مساله تاخیر هم در یک تصمیم گیری ضعیف موثر است که این خود دلیلی است بر اهمیت در نظر گرفتن تاخیر در شبکه.
روش ارائه شده باید مستقل و شفاف از دید برنامه کاربر و در نتیجه برنامه نویس باشد. به عبارت دیگر الگوریتم موازنه بار نباید به گونه ای باشد که برنامه نویس مجبور نباشد مستقیما در روال موازنه بار دخالت کند.
با توجه به توضیحات فوق می توان نتیجه گرفت که ویژگیهای ذکر شده تا حد زیادی به هم وابسته اند. به عنوان مثال تاخیراتی همچون تاخیر محاسبه[۷] و تاخیر ارتباط[۸] اثر نامطلوبی بر روی پایداری و در نتیجه قیاس پذیری الگوریتم می گذارد. با توجه به پارامترهای زیادی که در مساله موازنه بار دخیل هستند و همچنین متضاد بودن بعضی از ویژگیهای بیان شده، ارضای تمام ویژگیها در قالب یک الگوریتم واحد عملاً کار مشکل و یا حتی غیر ممکن است. اغلب روشهای موجود سعی در ارضاء یک یا چند هدف از اهداف فوق می نمایند.

---

[۱] Load sharing
[۲] Scalability
[۳] Adaptability
[٤] Stability
[٥] Application Transparency
[٦] Fault Tolerant
[۷] Computaion Delay
[۸] Communication Delay

۳-۲-۲- دسته بندی کلی روشهای موازنه بار موجود

به طور کلی روشهای موازنه بار به دسته های مرکزی و غیر مرکزی[۱]، ایستا یا پویا[۲]، دوره ای یا غیر دوره ای[۳] و نیز دارای حد آستانه و فاقد حد آستانه[٤] تقسیم بندی می شوند[۱۷، ۹۰، ۲۳] .

در الگوریتمهای مرکزی یک زمانبند مرکزی وجود دارد که تمام اطلاعات بار را از گره ها جمع آوری کرده و تصمیم مناسب را بر اساس این اطلاعات اتخاذ می نماید. این روشها معمولا برای محیط عظیمی مثل گرید قیاس پذیر نیستند [٤۲] در حالیکه ممکن است بعضی از آنها در سیستمهای توزیع شده _ به علت محدودیت تعداد گره ها، همگونی گره ها و فرکانس تغییرات کم آنها_کارایی خوبی ارائه دهند. این مساله یکی از تفاوتهای سیستمهای گرید و توزیع شده است. الگوریتمهای مرکزی به علت متکی بودن به یک گره، خیلی قابل اطمینان نیستند و از کار افتادن سیستم مرکزی معادل با از کار افتادن روال موازنه بار می باشد[٥]. با این حال الگوریتمهای مرکزی به لحاظ پیاده سازی بسیار ساده اند. در مدلهای غیر مرکزی (توزیع شده) معمولا یک یا چند گره خاص به نام سرور یا جمع آوری کننده وجود ندارد، بلکه تمام گره ها دارای اطلاعات تمام یا بخشی از گره های دیگر هستند از این رو هر گره به تنهایی می تواند در مورد موازنه بار تصمیم بگیرد. داشتن اطلاعات در مورد سایرین چه در روش مرکزی و چه در روش غیرمرکزی باعث ایجاد سربار زیاد در ارتباطات می شود. علاوه براین، در محیط گرید این اطلاعات برای مدت زیادی قابل اعتماد نبوده و باید با فرکانس بالایی به روز شوند.

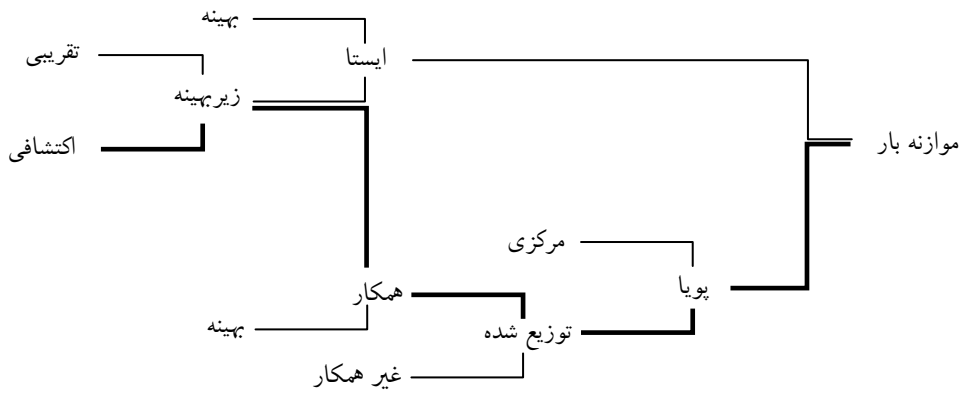
الگوریتمهای ایستا روشهایی هستند که تحت تاثیر شرایط سیستم نبوده و رفتار آنها از پیش تعیین شده است. در این الگوریتمها منابع مورد نیاز برنامه های کاربر در زمان ترجمه آنها و با توجه به دانش قبلی الگوریتم نسبت به سیستم تخصیص داده می شود. نمونه هایی از این الگوریتمها، الگوریتم انتخاب گره تصادفی[٦] و انتخاب گره به صورت نوبت گردشی[۷] [۵۳] است. در طرف دیگر، الگوریتمهای پویا تصمیماتشان را بر اساس شرایط سیستم اتخاذ می کنند. منظور از شرایط سیستم معمولا میزان بار پردازشی گره ها است که با معیارهای مختلفی همچون تعداد کارهای منتظر در صف آماده، نرخ ورود کار فعلی، میانگین زمان پاسخ و معیارهای مختلف دیگری که در مقالات مختلف استفاده شده اند، سنجیده می شود. البته کارایی این نوع الگوریتمها تا حد زیادی بستگی به معیاری که برای تخمین میزان بار انتخاب شده است دارد، ولی همانطور که گفته شد این معیار استاندارد خاصی نداشته و در الگوریتمهای مختلف از معیارهای مختلفی برای آن استفاده شده است، ولی در مجموع این روشها در محیطی مثل گرید کارایی و قیاس پذیری بهتری را به نسبت روشهای ایستا ارائه می دهند[٤۳].

بعضی از الگوریتمهای موازنه بار پویا، تطبیقی هستند[٤۳]. یک الگوریتم موازنه بار تطبیقی روشی پویاست که مطابق شرایط سیستم تغییرپذیر است. روشهایی که از این رویکرد استفاده می کنند می توانند عملیاتشان را بر اساس پس خور دریافتی از محیط تنظیم کنند.Casavants در [۸۵] روشهای موجود برای موازنه بار را دسته بندی کرده و نتیجه گرفت

---

[۱] Central & Decentral
[۲] Static & Dynamic
[۳] Periodic & Non-Periodic
[٤] With Threshold & Without Threshold
[٥] Single source of failure
[٦] Random
[۷] Round Robin

که الگوریتمهای پویای توزیع شده و همکار[۱]، موازنه بهتری را به نسبت سایر روشها موجب می شوند[٦٦،٥٨]. مسیر مربوط به این روشها در شکل۳-۱ پررنگ تر نشان داده شده است.



شکل۳-۱. دسته بندی روشهای موازنه بار .

از یک دید دیگر الگوریتمها را می توان به دو دسته دوره ای و غیر دوره ای تقسیم بندی کرد. در مدلهای دوره ای تبادل اطلاعات بین گره ها و یا موازنه بار بین آنها در فواصل زمانی مشخصی انجام می شود. این روشها باعث می شودکه همه گره ها در موعد زمانی خاص عملیات موازنه را انجام دهند و تبعیضی بین گره های مختلف - به لحاظ مورد موازنه قرار گرفتن- قائل نمی شوند. تعیین میزان فاصله زمانی در این روشها تاثیر به سزایی در کارایی الگوریتم دارد، پرواضح است که اهمیت ندادن به این مساله خیلی از اوقات باعث ارتباطات نابجا بین گره ها و یا باعث بروز نبودن اطلاعات و اتخاذ تصمیمات نابجا می شود. بالعکس، در بعضی از روشهای غیر دوره ای بعضی گره ها ممکن است برای مدت زمان زیادی مورد موازنه بار قرار نگیرند در حالیکه بعضی دیگر به طور دائم بارشان موازنه شود.

در یک تقسیم بندی دیگر می توان رویکردهای موجود برای موازنه بار در سیستمهای توزیع شده و گرید را به دو دستهٔ همکار[۲] و غیر همکار تقسیم بندی کرد[۱۱].

مدل همکار، همانطور که در شکل ۳-۱ دیده می شود زیر مجموعه ای از روشهای پویا است. در این روش چندین تصمیم گیرنده (مثل برنامه ها، کامپیوترها) وجود دارد که هر یک سعی در بهینه کردن عمل خود دارند. عاملهای تصمیم گیرنده آزادی کاملی در مورد نحوه ارتباطات و توافقهایشان[۳] با دیگران دارند. این شرایط قابل مدل سازی به صورت یک بازی دست جمعی است و روشهایی مثل "تئوری بازی[٤]" قالب مناسبی برای این نوع مدلسازی فراهم می کنند[۱۲].

در رویکرد "غیر همکار" هر کدام از کارها[٥] _ که تعداد آنها نامحدود فرض می شود_ به طور مستقل از دیگران سعی در کم کردن زمان پاسخ خود را دارد، و در نهایت همه آنها به تعادل (سکون) می رسند. این شرایط همانند یک بازی بدون همکاری بین کارهاست و به آن "موازنه وردراپ[٦]" [۲۵]گفته می شود.

به لحاظ اینکه درخواست موازنه بار از سوی چه گرهی ارسال می شود، الگوریتمهای کلاسیک موازنه بار به سه دسته راه اندازی شده توسط فرستنده[۱]، راه اندازی شده توسط گیرنده[۲] و ترکیبی[۳] تقسیم بندی می شوند [٤٨].

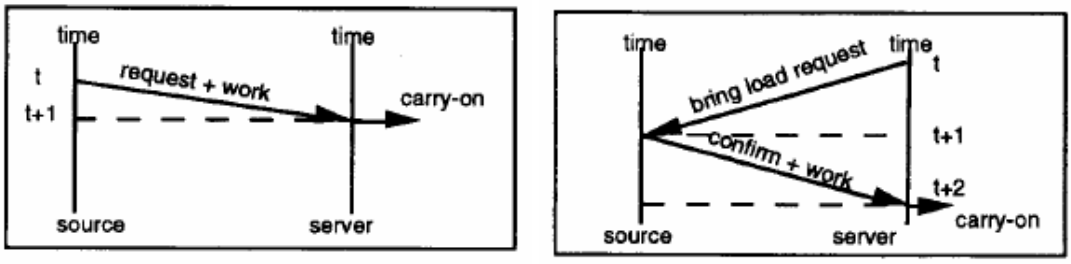
---

[۱] Distributed cooperative
[۲] Cooperative
[۳] Agreement
[٤] Game Theory
[٥] Jobs
[٦] Wardrop

در الگوریتمهای راه اندازی توسط فرستنده همانطور که در شکل۳-۲-الف دیده می شود، عملیات موازنه بار توسط گره پربار[۴] آغاز می شود. در اینصورت، یک گره پس از تشخیص پربار بودن به دنبال گرهی کم بار[۵] می گردد، در مرحله بعد با موافقت مقصد، گره مبدا اضافه بار خود را به گره کم بار ارسال می کند. البته انجام مرحله آخر اجباری نیست و می تواند حذف شود. عیب این روش این است که پردازش موازنه بار بایستی بر روی گرهی اجرا شود که خود دچار پرباری است.

در الگوریتمهای راه اندازی توسط گیرنده همانطور که در شکل۳-۲-ب دیده می شود، عملیات موازنه بار توسط گره کم بار آغاز می شود و گره کم بار به دنبال پردازشی برای اجرا بر روی خود، در گره های پربار می گردد. در اینصورت پس از موافقت گره پربار، عملیات ارسال بار به گره متقاضی بار انجام می شود. در این روش برعکس روش قبل پردازش موازنه بار بر روی گره کم بار اجرا می شود. مزیت دیگری که این روش دارد اینست که گره گیرنده _ که خود آغازگر موازنه بار بوده_ می داند چقدر بار جذب کند که خود، پربار نشود. در حالیکه در روش قبل ارضای این مساله با دشواری بیشتری همراه است.

در عین حال روش اول با یک تقاضا قابل انجام است در حالیکه روش دوم حتما نیاز به یک تقاضا و انتظار برای پاسخ مبدا دارد. در نتیجه همانطور که از شکل۳-۲ برمی آید مدل مبتنی بر فرستنده علیرغم سرعت بیشتر، قابلیت اطمینان کمتری دارد.



(الف) مبتنی بر گیرنده (ب) مبتنی بر فرستنده

شکل ۳-۲. دسته بندی روشهای موازنه بار[٦٦].

در مدل ترکیبی، هر گره می تواند بنا به شرایط خود در نقش راه اندازی توسط فرستنده یا راه اندازی توسط گیرنده ظاهر شود. در این الگوریتمها در شرایطی که یک گره پر بار باشد به صورت "راه اندازی توسط فرستنده" و در شرایطی که کم بار باشد به صورت "راه اندازی توسط گیرنده" موازنه بار را انجام می دهد.

۳-۲-۳- عملیات لازم برای انجام موازنه بار

به خاطر کارایی بهتر و کاربرد بیشتر الگوریتمهای پویا و با توجه به اینکه تمرکز اصلی این پایان نامه نیز بر روی همین دسته از الگوریتمها می باشد، در این قسمت برآنیم که روالهای عمومی در این دسته از الگوریتمهای موازنه بار را به تفصیل بیان کنیم.

---

[۱] Server Initiated
[۲] Receiver Initiated
[۳] Symmetric
[۴] Overloaded
[۵] Under-loaded

به طور کلی فرآیند موازنه بار در الگوریتمهای پویا دارای چهار روال اصلی زیر است:

الف) روال اندازه گیری بار[۱]

ب) روال تبادل اطلاعات[۲]

ج) روال راه اندازی[۳]

د) روال انجام عملیات نهایی موازنه بار

الف)روال اندازه گیری بار

عبارتست از بیان میزان بار پردازنده به صورتی که با سنگین تر شدن بار پردازشی پردازنده، این مقدار افزایش و با کاهش آن این مقدار کاهش یابد. چون این روال به دفعات و با بسامد بسیار زیادی در الگوریتمهای موازنه بار استفاده (اجرا) می شود، محاسبات مربوط به بدست آوردن آن بایستی تا حد امکان ساده و کارا باشد.

روال اندازه گیری بار – همانطور که در قسمتهای قبلی اشاره شد– نقش به سزایی در درست عمل کردن روال موازنه بار دارد. در واقع یکی از مشکلات الگوریتمهای فعلی نداشتن معیاری مناسب برای سنجش میزان بار می باشد. در اکثر الگوریتمهای موازنه بار فعلی پارامتری که به عنوان معیار سنجش میزان بار مورد استفاده قرار می گیرد "طول صف پردازشهای منتظر پردازنده" می باشد. اما این پارامتر نمی تواند معیار جامعی برای سنجش بار باشد. همانطور که می دانیم پردازشهایی که در سیستم اجرا می شوند، با تنگنای CPU[۴] و یا با تنگنای IO[۵] هستند. بنابراین صرف تعداد این پردازشها، نمی توان پربار یا کم بار بودن پردازنده را سنجید. چرا که ممکن است پردازنده ای دارای تعداد زیادی پردازش باشد، اما همه آنها رفتار IO از خود نشان دهند و بالعکس. در الگوریتمهای دیگر معیارهایی همچون نرخ ورود پردازش[۹۲] میانگین زمان تکمیل همه پردازشها و میانگین زمان پاسخ گره ها[۶] [٦] و یا ترکیبی از آنها [۹۲] به عنوان معیار سنجش بار مورد استفاده قرار گرفته اند.

مشکلی که در سنجش میزان بار وجود دارد ناشی از این واقعیت است که قبل از اجرای پردازشها نمی توان زمان لازم برای اجرای آنها را دانست. بنابراین اگر بتوان راهکاری برای پیشگویی صحیح میزان زمان لازم برای اجرای پردازشها در دست داشت، می توان میزان بار صحیح هر گره را یافت و دقت الگوریتمهای موازنه بار را افزایش داد.

ب) روال تبادل اطلاعات:

مشخص می کند که چگونه اطلاعات بار کاری لازم برای تصمیمات موازنه بار را جمع آوری و نگه داری کند. در واقع رابطه معکوسی بین هزینه جمع آوری اطلاعات سراسری بار گره ها و نگهداری وضعیت دقیق و بروز سیستم وجود دارد،که باید تعادلی بین آنها برقرار کرد. البته در محیط توزیع شده وسیعی همانند گرید، نگهداری اطلاعات کلیه گره ها امکان پذیر نیست و به ناچار باید اطلاعات زیر مجموعه ای از گره ها را نگهداری کنیم. باید در نظر داشت که در اینصورت دامنه ارسال بار هم به همان زیر مجموعه محدود خواهد شد. در مجموع سه روش پایه ای برای تبادل اطلاعات وجود دارد.

روش دوره ای، هر کدام از پردازنده ها وضعیت بار خود را به صورت دوره ای به اطلاع سایرین می رسانند. این عمل ممکن است حتی در شرایطی که این اطلاعات لازم نباشد نیز اتفاق بیافتد. مهترین مساله در این روش تعیین بازه زمانی

---

[۱] Load Measuring
[۲] Information Exchange
[۳] Initiation
[۴] CPU Bound
[۵] IO Bound
[۶] Mean Response Time

برای تبادل اطلاعات است. بازه های بسیار طولانی برای تبادل اطلاعات باعث تصمیم گیریهای غیر دقیق می شود، در حالیکه بازه های زمانی کوتاه حجم تبادلات را به شدت افزایش می دهد.

روش نیازی[۱] اطلاعات مورد نیاز بلافاصله قبل از انجام عملیات نهایی موازنه بار جمع آوری می شود. این روش حجم داده های مبادلاتی را تا حد زیادی کاهش می دهد، اما همراه با هزینه تاخیر اضافی برای عملیات موازنه بار است.Birman و Joseph [٦٩] به این نتیجه رسیدند که "موازنه بار وقتی امکان پذیر است که اطلاعات در موقع نیاز فورا در اختیار باشند". بنابراین روش نیازی چون در دسترس بودن فوری اطلاعات را برآورده نمی کند، خیلی مورد استقبال قرار نگرفت.

روش سوم، روش *تغییر حالت یا رویدادگرا*[۲] نام دارد که در آن با تغییر وضعیت هر گره اطلاعات جدید آن به سایرین ارسال می شود.

در نهایت ممکن است ترکیبی از این روشها نیز مورد استفاده قرار گیرند.به عنوان مثال Zhou [۸۳] به این نتیجه رسید که ترکیب روش دوره ای و رویدادگرا بهتر از سایر روشها عمل می کند. در اینصورت "حالت رویدادی" زمانی رخ می دهد که تغییرات ناگهانی و تعیین کننده ای در یک گره اتفاق بیافتد. در نتیجه این کار می توان اندازه دوره و همچنین حجم مبادلات بی دلیل را تا حد زیادی کاهش دهیم. در عین حال Zhou مدل دوره ای تطبیق پذیر[۳] را که در آن اندازه دوره منطبق بر شرایط سیستم تنظیم می شود را نیز پیشنهاد می دهد.

ج) روال راه اندازی:

این روال در مورد زمان شروع موازنه بار تصمیم گیری می کند. این تصمیم گیری باید همراه با سنجش نسبت کارایی به هزینه سربار تحمیلی باشد (یعنی انجام موازنه بار به صرفه باشد).

د) روال انجام عملیات موازنه بار (تصمیم گیری):

روشهای موازنه بار می کوشند اهدافی همچون کمینه سازی میانگین زمان پاسخ پردازشها یا بیشینه گی کارایی منابع را با اجرای پردازشها روی منابع توزیع شده به انجام برسانند. این هدف ممکن است در ابتدای ورود یک تقاضا یا پس از شروع به اجرای آن صورت گیرد. البته در هر حالت یک الگوریتم خوب و کارا بایستی هزینه مسیر را نیز در این بین در نظر داشته باشد.

این قسمت از موازنه بار نیز می تواند به صورت دوره ای، رویدادگرا و یا ترکیبی از این دو اجرا شود. در هر صورت این روال در عملکرد خود نیاز به سه قانون پایه ای دارد: قانون محلیت[٤]، قانون توزیع[٥] و قانون انتخاب[٦].

قانون محلیت بیان می کند چه دامنه ای از پردازنده ها (منابع) مشمول عمیات موازنه بار می شوند. این دامنه ممکن است محلی، یعنی درون هر گره و یا سراسری، یعنی بین گره های مختلف گرید باشد.

قانون توزیع، مشخص کننده چگونگی موازنه بار بین منابع موجود در دامنه است. موازنه بار ممکن است در هنگام ورود کار به سیستم و توسط خود زمانبند انجام شود و یا ممکن است پس از شروع اجرا، توسط الگوریتمهای خاص موازنه بار مجددا به سایر گره ها ارسال شود. مسلما حالت دوم بایستی توسط فرآیندی مجزا از فرایندهای خود سیستم انجام شود.

---

[۱] On Demand
[۲] Event-Oriented
[۳] Adaptive Periodic
[٤] Locality
[٥] Distribution
[٦] Selection

قانون انتخاب نیز مشخص کننده این امر است که عملیات موازنه بار به صورت پس گرفتنی انجام شود یاخیر. در روش پس گرفتنی ممکن است پردازش در حین اجرا به سایر گره ها انتقال یابد که در اینصورت زیر ساخت مورد استفاده بایستی دارای قابلیت تحرک قوی باشد. در روش غیر قابل پس گرفتنی فقط پردازشهایی را می توان ارسال کرد که هنوز زمانی از پردازنده نگرفته اند.

۳-۳- بررسی روشهای پیشین موازنه بار در گرید و سیستمهای توزیع شده

تا کنون روشهای زیادی برای موازنه بار، به خصوص در سیستمهای توزیع شده ارائه شده است. همه این روشها دارای مشخصات بیان شده در قسمت ۳-۲-۳ می باشند و هر کدام از آنها سعی در ارضا یک یا چند هدف از اهداف و ویژگیهای بیان شده در قسمت ۳-۲-۱ را دارند. البته همانطور که گفته شد، برآوردن همه اهداف ذکر شده _ به خاطر تضاد ذاتی آنها _ در یک الگوریتم امکان پذیر نیست. بنابراین بایستی بنا به شرایط و اهداف سیستم به بعضی از ویژگیها اهمیت بیشتری داده و الگوریتم ارائه شده را با جهت گیری آن اهداف طراحی نمود.

به خاطر جدید بودن مبحث گرید، تا کنون الگوریتمهای زیادی برای موازنه بار ارائه نشده است: از طرفی باید توجه داشت که گرید به لحاظ شرایط محیطی و ساختاری، تفاوت عمده ای با پیشینیان خود یعنی سیستمهای توزیع شده دارد، در نتیجه نمی توان از الگوریتمهای موجود موازنه بار برای سیستمهای توزیع شده در گریدهای محاسباتی استفاده کرد. یکی از مهمترین تفاوتهای اساسی گرید وسعت بی حد آن است: همانطور که گفته شد گرید را می توان سیستم توزیع شده ای دانست که هر یک از گره های آن خود می تواند یک سیستم چند پردازنده ای (MP)، یک کامپیوتر شخصی و یا حتی یک سیستم توزیع شده باشد. این گستردگی سبب شده است که مجبور به استفاد ه از چندین سطح موازنه بار باشیم. به عنوان مثال موازنه بار در سطح هر گره-که خود می تواند با استفاده از روشهای موازنه بار در سیستمهای توزیع شده یا چند پردازنده ای انجام شود- و یا موازنه بار در سطح کل گرید که به انتقال بار بین گره های مختلف گرید می پردازد.

در این قسمت سعی در تشریح الگوریتمهای پایه ای موازنه بار در محیط گرید و سیستمهای توزیع شده می کنیم و نکات قوت و ضعف هر یک را بررسی می کنیم. در این ضمن توجه ویژه ای به روشهای موازنه بار ارائه شده در ARMS داریم.

۳-۳-۱- بررسی کلی روشهای موجود

همانطور که گفته شد موازنه بار هم در زمان ایجاد و مکان یابی[۱] پردازش می تواند انجام شود و هم پس از شروع به اجرای پردازش[۲]. در روشهایی که موازنه بار در زمان ایجاد (ورود) پردازش انجام می شود، معمولا وظیفه الگوریتم موازنه بار به صورت جزئی از سیاست زمانبندی در می آید. اما در روشهایی که پس از شروع به اجرای پردازش موازنه بار انجام می شود، الگوریتم موازنه بار عضو مستقلی در سیستم مدیریت منابع است.

در واقع می توان مساله موازنه بار را نوعی از مسائل بهینه سازی دانست که در آن هدف تابع بهینه ساز ممکن است کم کردن میانگین زمان پاسخ کلی سیستم (به ویژه در سیستمهای مبتنی بر کاربر که معمولا به صورت محاوره ای یا

[۱] Allocation
[۲] Start up

بلادرنگ هستند) و یا افزایش بهره وری منابع سیستم (به ویژه در در سیستمهای مبتنی بر منبع که معمولا برای برنامه های دسته ای کاربرد دارد)باشد. در گریدهای محاسباتی چون گره های مختلف، متعلق به سازمانهای مختلفی می باشند و بعضا ممکن است این سازمانها اهداف تجاری داشته باشند، علاوه بر موارد فوق، افزایش سود دهی نیز می تواند هدفی برای الگوریتم موازنه بار باشد.

مسئله موازنه بار همچون بسیاری از مسائل بهینه سازی جزء مسائل رام نشدنی[١] محسوب می شود [٣٣]. یعنی نمی توان در یک زمان مورد قبول بهترین جواب ممکن را یافت. بنابراین باید به دنبال جوابهای نسبتا خوب (و نه الزاماً بهترین) بود. ازطرفی زمان رسیدن به جواب هم در الگوریتم موازنه بار اهمیت ویژه ای دارد. به عبارت دیگر الگوریتم موازنه بار نباید خیلی زمانبر باشد و نباید خود به باری سنگین برای گره های محاسباتی تبدیل شود. این مساله به ویژه در روشهایی که در سطح زمانبند عمل می کنند اهمیت بیشتری دارد. بنابراین برای رسیدن به جوابهای نسبتا خوب در یک زمان قابل قبول در مسائل بهینه سازی بهتر است از روشهای تقریب زننده و اکتشافی[٢] استفاده کنیم. بر همین اساس روشهای موازنه بار موجود از روشهای اکتشافی همانند الگوریتمهای ژنتیک[٣] [٤،٤٠]، شبکه های عصبی [١٥]، منطق فازی[٣]، استفاده از عاملهای هوشمند[٤] [٤٢]، روشهای تقریبی [٩١] و روشهای هوش جمعی مثل روشهای مبتنی بر دسته مورچه ها[٥] [٢] استفاده می کنند.

با توجه به اینکه بر روی هر یک از روشهای تقریب زننده فوق چندین کار تحقیقاتی و دانشگاهی انجام شده است، به منظور خلاصه سازی در ادامه این بخش سعی می کنیم بر روی آن دسته از کارهایی را که به نظر می آید با مشخصات محیط گرید انطباق بیشتری دارند تمرکز کنیم.

٣-٣-٢- استفاده از الگوریتمهای ژنتیک

یک روش اکتشافی کارا در موازنه بار، استفاده از روشهای مبتنی بر الگوریتم ژنتیک است. A.Zomaya و همکارانش در [٤] مبانی استفاده از روشهای الگوریتم ژنتیک را در موازنه بار مورد بررسی قرار داده و به این نتیجه رسیدند که الگوریتمهای ژنتیک در مواقعی که حتی سایر روشهای اکتشافی کارایی خود را از دست می دهند نیز کارایی خوبی از خود نشان می دهند. البته وی این کار را در یک سیستم توزیع شده انجام داده است و به علت ضعف در قیاس پذیری الگوریتمهای ژنتیک نمی توان آن را در سطح کل گرید تعمیم داد.

بر همین اساس J.Cao و همکارانش در [٤٠] روش موازنه باری درARMS ارائه دادند. این الگوریتم موازنه بار را در سطح گره و توسط زمانبند انجام می دهد: یعنی سعی در پخش کردن کارهای تحویل داده شده به یک گره می کند به صورتیکه تمام پردازشهای محوله بتوانند قبل از پایان یافتن مهلت زمانیشان[٦]، اجرایشان را به پایان برسانند. در واقع زمانبند قصد دارد در عین توجه به مهلت زمانی پردازش منتظر، زمان بیکاری پردازنده های هر گره را هم به حداقل برساند. به عبارت دیگر سعی می شود یک ترتیب بهینه برای اجرای پردازشهای منتظر اجرا در یک گره پیدا شود، که در پیدا کردن این ترتیب بهینه اجرا از الگوریتمهای ژنتیک استفاده شده است. همچنین از قابلیتهای PACE نیز در این روش مورد استفاده شده است. روش کار این الگوریتم به طور خلاصه در اینجا ذکر می شود.

---

[١] NP Complete
[٢] Heuristic
[٣] Genetic Algorithm
[٤] Intelligent Agents
[٥] Ant Colony
[٦] Deadline

همانطور که گفته شد یک گره P در گرید را به صورت کلی می توان یک مجموعه پردازنده تصور کرد. این مجموعه را می توان به صورت زیر مدل کرد:

$$P : \{p_i \mid i = 1,...,n\} \quad (۱-۳)$$

توانایی (کارایی) هر یک از پردازنده های[۱] گره P را که با استفاده از PACE بدست می آید را نیز می توان به صورت زیر مدل کرد:

$$\rho : \{\rho_i \mid i = 1,...,n\} \quad (۲-۳)$$

مجموعه ای از کارهای موازی را می توان به صورت مجموعه T که در زیر آمده است در نظر گرفت:

$$T : \{T_j \mid j = 1,...,m\} \quad (۳-۳)$$

همچنین اطلاعات مربوط به نیازهای کارآیی[۲] هر یک از این برنامه ها را نیز می توان به صورت مجموعه زیر بیان کرد:

$$\sigma : \{\sigma_j \mid j = 1,...,m\} \quad (٤-۳)$$

این مجموعه نیز توسط ابزار AT در PACE قابل حصول است.

چون در ARMS اولیه فقط برنامه های بلادرنگ پشتیبانی می شدند، هر برنامه علاوه بر نیاز کارایی خود بایستی مهلت زمانی مورد نیاز خود را نیز بیان می کرد. بنابراین برای هر یک از اعضای مجموعه T مهلت زمانی خاصی به صورت آنچه در مجموعه زیر آمده است، نیاز بود:

$$\delta : \{\delta_j \mid j = 1,...,m\} \quad (۵-۳)$$

در این صورت زمان تکمیل پردازش j عبارت بود از:

$$\eta_j = \tau_j + t_x(\overline{\rho}_j, \sigma_j) \quad (٦-۳)$$

که در آن:

η_j: زمان تکمیل پردازش j

τ_j: زمان شروع پردازش.

$t_x(\overline{\rho}_j, \sigma_j)$: تابعی است که زمان اجرای هر پردازش j با مدل کارایی بصورت σ_j و بر روی پردازنده ای با توانایی $\overline{\rho}_j$ را بدست می دهد. حاصل این تابع توسط موتور ارزیابی[۳] در PACE بدست می آید.

بنابراین بایستی حداکثر زمان اجرا، حداقل شود و همچنین هر پردازش در قبل از موعد زمانی خود به اجرا برسد، یعنی:

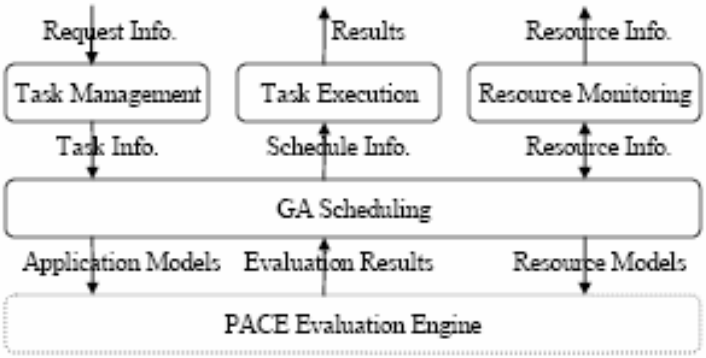
$$\omega = Max\{\eta_j\} \quad (۷-۳)$$
$$1 \le j \le M$$

ω بیان کننده این است که در یک روش (ترتیب) خاص اجرای پردازشها (یک ترتیب زمانبندی خاص)، دیرترین زمان اجرا چیست. هدف، حداقل کردن میزان ω است و همچنین برقراری $\eta_j \le \delta_j$ (اجرا شدن در زمان مقرر) است.پس مساله تبدیل به یک مساله بهینه سازی ترکیبی (ترکیبی از دو فاکتور فوق) می شود و هدف نهایی هم پیدا کردن یک ترتیب اجرای پردازشها بر روی یک گره گرید است که این خواص را برآورده سازد. برای این منظور از یک الگوریتم ژنتیک استفاده شده است. در این صورت مجموعه زمانبندیهای مختلف ایجاد و راه حلی که خاصیت مورد نظر را بیش از سایرین برآورده می کند انتخاب می شود.

---

[۱] Resource Model
[۲] Application Model
[۳] Evaluation Engine

همانطور که دیده می شود هدف اصلی این الگوریتم موازنه بار نمی باشد بلکه هدف پیدا کردن یک ترتیب اجرای بهینه برای اجرا بر روی یک گره است. معماری کلی سیستم به صورت شکل زیر است:



شکل۳-۳. معماری الگوریتم در قسمت مدیریت محلی ARMS [٤٠].

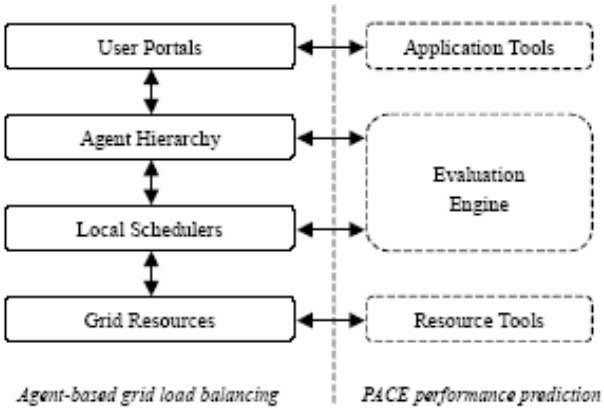
Task Management: تقاضا ها به این قسمت وارد شده وبرای زمانبندی و اجرا به صف می شوند تا توسط الگوریتم ژنتیک مورد پردازش قرار گیرند.

Resource Monitoring: وظیفه جمع آوری اطلاعات آماری مربوط به پردازنده هایی که کارها قرار است بر روی آنها اجرا شوند را دارد.

Task Execution: وظیفه اجرای برنامه مرتبط با یک برنامه را بر روی لیست پردازنده های زمان بندی شده را دارد.

PACE Evaluation Engine: استحصال داده های پیشگویانه[١] در مورد زمان اجرای پردازش قبل از اجرای آن با استفاده از مدل برنامه (که بیان کننده نیازهای پردازش بوده و توسط PACE بدست می آید) و مدل منبع (که بیان کننده توانائیهای منبع بوده و توسط PACE بدست می آید) را به عهده دارد.

معماری زیر بیان کننده جایگاه الگوریتم زمانبندی در کل سیستم ARMS است.



شکل۳-٤ . جایگاه زمانبند هر گره در کل معماری ARMS [٤٠].

[١] predictive

نتیجه گیری:

همانطور که دیده می شود و طبق نتایج موجود در مقاله این روش تاثیر مثبتی بر کارایی هر گره دارد. مزیت دیگر این روش این است که در سطح گره می باشد، یعنی برای اجرای آن احتیاج به تحمیل پردازش اضافه ای به سیستم مدیریت منابع نیست و خود زمانبند کار موازنه بار را نیز انجام می دهد.

چون الگوریتم ژنتیک قیاس پذیر نمی باشد، این روش قابل تعمیم به کل گرید نیست و تنها می تواند در سطح هر یک از گره ها اجرا شود.

۳-۳-۳- استفاده از هوش جمعی

هوش گروهی روشهایی اکتشافی در حل مسائل بهینه سازی هستند که از رفتار حشرات اجتماعی مثل مورچه ها، زنبورها و... الهام گرفته شده است. در این روشها هر یک از عناصر (که معمولا یک عامل سیار در نظر گرفته می شود) دارای هوش کمی نسبت به محیط و عملکرد خود هستند، اما تعامل انبوهی از این عاملهای کم هوش می تواند مسائل پیچیده ای را حل کند. برای مثال مورچه ها دارای هوش کمی نسبت به دشمنان و تغییرات محیط خود هستند. مورچه ها عملیاتی مثل سازمان دهی جسدها و جستجو برای غذا را با استفاده از این هوش گروهی انجام می دهند. مورچه ها این عملیات را بر اساس ارتباط غیر مستقیم که به آن stigmergy گفته می شود انجام می دهند. این ارتباط از طریق ماده اسیدی به جا مانده از مورچه ها - که به آن اسید مورچه[۱] گفته می شود- انجام می شود. هر کدام از مورچه های خود مختار، محیط خود را بنا به شرایط خود تغییر می دهند. ارتباط مورچه ها از طریق همین محیط آغشته به اسید انجام می شود.

این ایده اولین بار توسط Dorigo در [٥٦] به عنوان یک رویکرد چند عامله[۲] برای حل مسائل بهینه سازی ترکیبی[۳] ارائه شد. امروزه الهام از اینگونه عملیات در حل بسیاری از مسایل بهینه سازی مثل مسیریابی وسایل نقلیه [٧٦]، شبکه های کامپیوتری راه دور [٤٦]، هماهنگی روبوتها [٧١]، مرتب سازی [٤٣] و به خصوص موازنه بار [٤٢،٣٨،٦٧] مورد استفاده قرار گرفته است.

یکی از ویژگیهای منحصر به فرد روشهای بهینه سازی مبتنی بر مورچه، "خود سازماندهی[٤]" می باشد. یعنی مورچه ها در انجام کار خود احتیاج به مدیریت خاصی نداشته و در صورت بروز خطا، تا حد امکان آن را خود به خود ترمیم می کنند. همانطور که قبلا نیز گفته شد، مدیریت منابع در گرید بایستی تا حد ممکن خودسامانده باشد. از این رو روشهای هوش جمعی(مبتنی بر مورچه) می تواند روش مناسبی برای پیاده سازی این هدف در قسمتهای مختلف سیستم مدیریت منابع گرید باشد. از جمله در مساله موازنه بار نیز می توان از این روشها استفاده کرد. در همین رابطه، یک الگوریتم پایه ای برای موازنه بار در گرید به نام Messor توسط Montressor و همکارانش ارائه شده است [۲]. به خاطر اهمیت این روش و استفاده از آن در این پایان نامه، به بررسی کاملتر آن می پردازیم.

Messor یک سیستم گرید است که هدف آن اجرای همروند پردازشهای موازیی است که امکان شکسته شدن به چندین برنامه مستقل کوچکتر را دارند. Messor بر روی بستر AntHill [٦٧] _که یک محیط چند عامله است_ پیاده سازی شده است. در سیستم Messor از ایده Resnick [٦١] برای موازنه بار استفاده شده است. ایده Resnick با

---

[۱] Pheromone
[۲] Multi Agent
[۳] Combinatorial Optimization
[٤] Self Organizing

الهام از عملیات مورچه ها در تمیز کردن لانه و دسته بندی اجساد پخش شده درسطح آن در قالب دسته های منظم بود. این کار بویژه توسط نوع خاصی از مورچه ها به نام Messor Sancta [١٦] انجام می شود. با مشاهده دسته های منظم ایجاد شده توسط مورچه ها ممکن است به اشتباه فکر کنیم که عملیات تمیز کردن سطح لانه با استفاده از نوعی مورچه "فرمانده" که به کار سایرین نظارت داشته انجام شده است، اما در واقع اینچنین نیست بلکه آنچه حاصل شده، به صورت خودسازمانده و با هوش گروهی به انجام رسیده است. Resnick همچنین به این نتیجه رسید که این رفتار مورچه ها قابل شبیه سازی است. برای این شبیه سازی کافیست مورچه های مصنوعی از سه قانون زیر پیروی کنند:

(i) یک مورچه به صورت تصادفی حرکت کند تا به یک شیی برخورد کند.

(ii) وقتی مورچه ای در حال حمل شیئی می باشد، شیئ را رها کرده و به حرکت تصادفی (نامشخص) خود ادامه دهد.

(iii) اگر مورچه در حال حمل شیئی نیست، شئ را برداشته و به حرکت تصادفی خود ادامه دهد.

علی رغم این عملکرد ساده، دسته ای از مورچه ها که با این الگو رفتار کنند می توانند توزیع اولیه اشیاء را تبدیل به دسته های تفکیک شده منظم کنند.

اگر رفتار مورچه ها به صورت معکوس تغییر کند، نتیجه مورد انتظار عکس عملیات فوق است. یعنی مورچه ها سعی در پخش اشیاء متراکم شده در محیط خود می کنند. در این صورت قوانین فوق باید به صورت زیر تغییر کنند:

(i) وقتی مورچه ای شیئی حمل نمی کند، حرکت تصادفی اش را تا زمانیکه به یک شیئ برخورد کند، ادامه می دهد.

(ii) وقتی مورچه ای شیئی حمل می کند، آنرا فقط در صورتی رها می کند که برای مدت زمانی در حال حرکت باشد، اما به شیئی برخورد نکرده باشد.

الگوریتم موازنه بار Messor نیز از همین قوانین تبعیت می کند. در اینصورت اشیائی که حمل می شوند همان کارهای واقعی هستند. مورچه ها در طول حیات خود در این سیستم می توانند در حالتهای *Search–Max* یا *Search–Min* قرار گیرند. در حالت *Search–Max* یک مورچه به صورت تصادفی حرکت می کند تا یک گره پر بار پیدا کند. سپس مورچه به حالت *Search–Min* می رود تا یک گره کم بار را پیدا کند. مفهوم بار در این روش "تعداد عناصر موجود در صف هر گره" بیان شده است. پس از این مراحل، مورچه عمل موازنه بار بین دو گره پیدا شده را انجام می دهد. وقتی یک مورچه در حین حرکت خود به یک گره می رسد، اطلاعات گره هایی را که تا آن زمان ملاقات کرده را برای استفاده سایر مورچه هایی که از آن گره عبور می کنند باقی می گذارد. سایر مورچه ها می توانند از این اطلاعات برای عملکرد بهتر خود استفاده کنند.

نتیجه گیری:

جدا از اینکه روش ارائه شده راه جدیدی را پیش روی محققین در زمینه موازنه بار باز کرد، نقائصی در آن وجود دارد که در اینجا به آنها اشاره می شود. اول اینکه همانطور که قبلا بحث شد، تعداد پردازشهای موجود در صف، مقیاس کاملی برای سنجش میزان بار نیست. دوم اینکه به علت به پویایی محیط گرید اطلاعات به جای مانده از مورچه ها در هر گره ها نمی توانند برای مدت زیادی قابل اعتماد باشد و ممکن است باعث تصمیم گیریهای نابجایی برای سایر مورچه هایی باشد که از آن گره عبور می کنند. همچنین تعداد مورچه ها در این سیستم ثابت است یعنی چه سیستم به حالت بالانس نزدیک باشد وچه در شرایط عدم توازن حاد باشد، تعداد مورچه ها ثابتند.

با توجه به نتایج خوب Messor و همچنین سازگاری ساختار مورچه ها با محیطهای چند عامله، منطقی است که از آن برای موازنه بار بین گره های گرید در مدل مدیریتی ARMS نیز استفاده کنیم. بر همین اساس J.Cao بر اساس عملکرد Messor راهکاری را برای موازنه بار بین گره ها درARMS ارائه داده است [٦٧]. توضیح نحوه کار این الگوریتم احتیاج به ذکر مواردی در رابطه با ARMS دارد که در زیر ارائه شده است.

در فصل ۱ بیان شد که سیستم ARMS اولیه فقط قابلیت پشتیبانی از برنامه های بلادرنگ را داشت. به عبارت دیگر می توان گفت که این سیستم مبتنی بر کاربر بود و سعی در برآورده ساختن نیاز کاربر در اسرع وقت داشت. اما همه برنامه هایی که به گرید وارد می شوند الزاما به صورت بلادرنگ نیستند بلکه در بسیاری از آنها فقط کاربر می خواهد که برنامه اش تحت گرید اجرا شود (این امر ممکن است به علت پیچیده گی برنامه کاربر و یا احتیاج به ماشینی با معماری خاص باشد). یعنی ممکن است برنامه های ورودی به گرید رفتارهای پردازشهای دسته ای را از خود نشان دهند. در این نوع از تقاضاها دیگر هدف تامین نیاز کاربر در اسرع وقت نیست؛ بلکه هدف استفاده بهینه و با کارایی بالا از منابع موجود در گرید است. در اینصورت به لحاظ تجاری سرمایه گذاران و صاحبان منابع سود بیشتری خواهند برد و از منابعشان به نحو مطلوبتری استفاده خواهد شد. مسلما در این موارد، چون عجله ای برای اجرا نداریم و هدف هم افزایش کارایی منابع است الگوریتمهای موازنه بار کاربرد بیشتری دارند. در این صورت به احتمال زیاد میانگین زمان انتظار درخواستها نیز کاهش می یابد. اما چون زمانبند ARMS اولیه راهکاری برای زمانبندی پردازشهای دسته ای نداشت، پیش از هر چیز نیاز به ارائه یک الگوریتم زمانبندی کارا برای ایجاد قابلیت پشتیبانی از پردازشهای دسته ای و بلادرنگ را به طور همزمان بود.

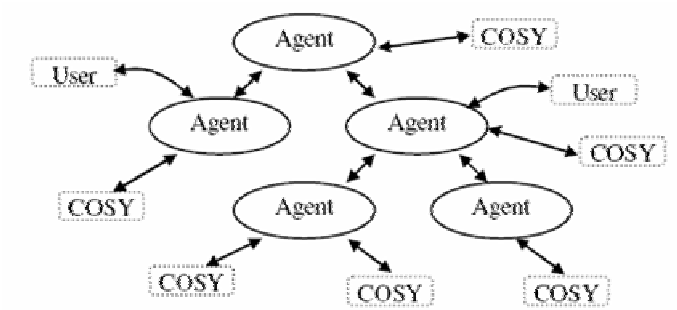
COSY [۲۷، ۳۹] یک زمانبند کارهای دسته ای[۱] است. این زمانبند علاوه بر امکان زمانبندی پروسسهای دسته ای، دارای امکان اضافه تری به نام رزرواسیون پیشرفته[۲] نیز می باشد. رزرواسیون پیشرفته به این معنی است که زمانبند قدرت برآوردن محدودیتهای مورد نیاز برنامه کاربر را دارد. این محدودیت ممکن است شروع اجرای یک برنامه خاص بر روی یک یا چند پردازنده خاص در زمان خاص و یا پایان یافتن اجرا در یک موعد زمانی باشد. البته باید توجه کرد که پشتیبانی از رزرواسیون پیشرفته باعث افزایش طول صف کارها و کم شدن کارایی برای کارهای عادی می شود؛ ولی به هر جهت برای انعطاف بخشیدن به سیستم مدیریت منابع مجبور به تحمل این سربار هستیم.

روش عملکرد COSY به صورت صف چند سطحی است که در هر صف می توانیم اولویتها و محدودیتهای خاص آن صف را داشته باشیم. مثلا ممکن است دو صف جداگانه "شب" و "روز" داشته باشیم، در صف شب حداکثر زمان استفاده از پردازنده را چهار ساعت و در صف روز این زمان را یک ساعت معین کنیم. در هر صف از روش ترتیبی[۳] برای نوبت دهی به پروسسهای منتظر استفاده می شود. روش COSY همچنین قابلیت اولویت دهی به کاربران را نیز دارد و کاربران بر اساس اولویتشان در صفهای مختلف قرار می گیرند. همچنین این امکان وجود دارد که یک کاربر با یک اولویت خاص، در صفهای مختلف اولویتهای مختلفی داشته باشد. محدودیتی که این سیستم دارد این است که هر کاربر در هر لحظه می تواند چندین تقاضای دسته ای و فقط یک تقاضای محاوره ای داشته باشد.

زمانبند COSY بعدها به ساختار ARMS اضافه شد [۳۹] تا به آن قابلیت پشتیبانی از کارهای دسته ای و بلادرنگ را _ به صورت همزمان_ بدهد. همانطور که در شکل ۳-۵ دیده می شود، با این تغییر هر عاملARMS ممکن است دارای یک یا چند زمانبند COSY باشد. وجود چند زمانبند COSY به این علت است که اکثر گره های گرید به

[۱] Batch Job Scheduler
[۲] Advance Reservation
[۳] FCFS

صورت چند پردازنده ای یا کلاستر می باشند، در این صورت برای هر پردازنده یا هر دسته پردازنده یک زمانبند COSY مجزا در نظر گرفته شده است. با ترکیب COSY و عاملهایARMS از این پس عاملها اطلاعات منابع خود را از COSY پرس و جو و دریافت می کنند(قسمت مدیریت محلی).



شکل۳-۵. عاملهای ARMS به همراه COSY[٤٢].

با وجود زمانبند COSY و اضافه شدن قابلیت پشتیبانی برنامه های دسته ای به ARMS و با توجه به ضرورت استفاده از الگوریتم موازنه بار برای جابجائی پردازشهای دسته ای و نیل به بهره وری بالاتر منابع گرید، در [۳۸] با استفاده از تئوری هوش جمعی و استفاده از مدل مورچه ها راهکاری برای موازنه بار پردازشهای دسته ای در ARMS ارائه شده است که در زیر این راهکار مورد نقد و بررسی قرار می گیرد.

الگوریتم مبتنی بر مورچه های ارائه شده در ARMS تا حد زیادی مشابه روش Messor بوده و به صورت زیر عمل می کند:

۱. یک مورچه از یک عامل به عامل دیگر رفته (به صورت تصادفی) و سعی می کند مشخصات عاملی که بار بیشتری دارد را به خاطر بسپارد.

۲. پس از پیمودن m گام و پیدا کردن عاملی با بیشترین بار در طی این گامها، مورچه تغییر وضعیت داده و این بار سعی در به خاطر سپردن مشخصات عاملی با کمترین بار می کند.

۳. پس از پیمودن m گام و پیدا کردن عاملی با کمترین بار، مورچه یک گام توقف کرده تا پیشنهاد موازنه بار بین عامل حاوی بیشترین بار و عامل حاوی کمترین بار را بدهد.

٤. پس از اجرای موازنه بار، حلقه از مرحله ۱ دوباره تکرار می شود.

در هر گام مورچه عامل بعدی را به طور تصادفی از بین همسایگان خود انتخاب می کند. البته یک بهینه سازی انجام شده در این زمینه این است که مورچه به جای حرکت تصادفی، هر بار همسایه ای با بیشترین بار را انتخاب کند. پس از شروع، مبادله تا زمانیکه میانگین طول صفهای هر دو گره پربار و کم بار یکسان شود ادامه می یابد. به لحاظ پیاده سازی، مورچه را می توان به صورت یک فایل XML که بین عاملها مبادله می شود در نظر گرفت.

همانند روش قبلی در این الگوریتم نیز میانگین طول تمام صفهای COSY در یک عامل به عنوان معیار سنجش بار در نظر گرفته شده است.

یکی از عوامل موثر در سریعتر شدن همگرایی روش، افزایش تعداد مورچه ها (n) است. البته در اینصورت هزینه ارتباطات[۱] بین گره هابه شدت افزایش می یابد. فاکتور تاثیرگذار دیگر در کارایی این روش، تعداد گامهای مورچه(m) می باشد. اگر تعداد گامهای مورچه ها کم باشد، تعداد دفعاتی که موازنه بار انجام می شود افزایش می یابد، در حالیکه

[۱] Communication

تعداد گامهای زیاد باعث کم شدن تعداد دفعات عملیات موازنه بار می شود. بنابراین می توان نتیجه گرفت که برای شرایط مختلف سیستم (سطوح مختلف عدم توازن)، تعداد گامهای متفاوتی نیاز است. مثلا در هنگامی که سیستم به شدت نامتوازن[1] است احتیاج به تعداد عملیات موازنه بار (دفعات بالانس) بیشتری است. یعنی m بایدکوچک باشد. اما در شرایط نزدیک به حالت تعادل، احتیاج کمتری به عملیات موازنه بار است، در نتیجه مقادیر m بزرگتر باعث کارایی بهتر می شوند.

نتیجه گیری:

همانطور که دیده می شود این روش بر خلاف روش اول (که در سطح گره بود)، در سطح کل گرید (بین گره ها) انجام می شود و منجر به موازنه بارخوبی در سطح کل گرید می شود. اما چون این روش به صورت یک پروسس جدا بر روی گرید اجرا شده و احتیاج به ارتباطات زیادی _ به خصوص در هنگامیکه تعداد مورچه ها زیاد باشد_ دارد، سربار زیادی برای سیستم به همراه دارد.

مشکل دیگر این روش این است که مورچه ها به طور عادلانه ای به گره ها سرکشی نمی کنند. به عبارت دیگر ممکن است بعضی گره ها برای مدت زیادی توسط هیچ مورچه ای پیمایش نشوند، در حالیکه بعضی دیگر دائم توسط مورچه ها ملاقات شوند. این مشکل در آزمایشاتی که در [۳۸] انجام شده مشخص است.

مشکل دیگر عدم انعطاف و تطبیق پذیری این الگوریتم می باشد. همانطور که دیده شد برای نیل به کارایی مطلوب، بایستی تعداد مورچه ها و تعداد گامهای آنها مطابق با شرایط سیستم مشخص شود، در حالیکه در الگوریتم اولیه تعداد مورچه ها و تعداد گامهای آنها توسط کاربر مشخص می شوند و در طی زمان موازنه بار مقدار آنها بدون تغییر است؛ البته این کار در سیستم مدیریت منابع گرید که یکی از اهداف اصلی آن ایجاد شفافیت برای کاربر است به هیچ وجه منطقی به نظر نمی رسد. در یک روش بهینه می توان تعداد مورچه ها و گامهایشان را منطبق بر شرایط سیستم تنظیم کرد.

نقص دیگری که در این الگوریتم می توان به آن اشاره کرد، عدم استفاده بهینه مورچه از فرصتهاست. همانطور که در قسمت قبل دیده شد، هر مورچه در هر دوره کاری خود $2m+1$ پرش انجام می دهد، اما پس از آن فقط بار دو گره را با هم موازنه می کند. در یک روش بهینه می توان از این فرصت استفاده بهتری برده و در پایان گامها تعداد بیشتری گره را در موازنه دخالت داد.

۳-۳-٤- روشهای سطح زمانبند

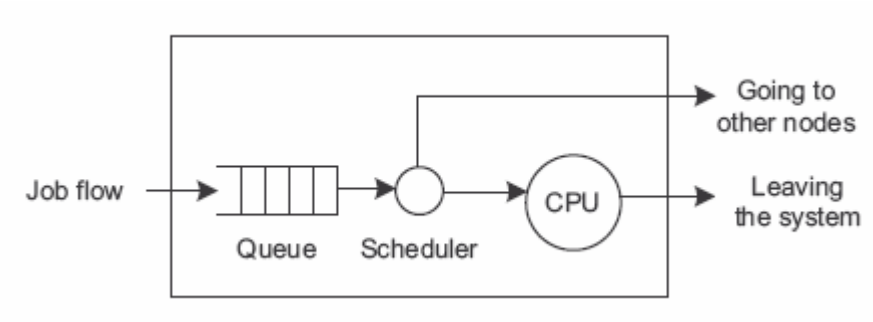
همانطور که قبلا گفته شد، دسته ای از روشهای موازنه بار توسط زمانبند اجرا می شوند، یعنی خود زمانبند در هنگام ورود برنامه کاربر سعی می کند آنرا در گره مناسبی جای دهد. این روشها به علت اینکه بار اضافی بابت خود پردازش موازنه بار به سیستم تحمیل نمی کنند مورد توجه اند. اغلب الگوریتمهای موازنه بار ایستا مانند الگوریتم موازنه بار تصادفی و الگوریتم موازنه بار نوبت گردشی به همین گونه اند. بعضی از الگوریتمهای پویا نیز به همین شکل پیاده سازی می شوند. در این الگوریتمها سعی شده هوشمندی لازم برای موازنه بار به زمانبند هر گره منتقل شود، به صورتیکه خود زمانبند بر روی صف کنترل داشته و در صورت بروز شرایط خاصی که مبین احتمال بروز عدم توازن بار در سیستم است_ با توجه به اطلاعاتی که از گره های همجوار خود دارد_ موازنه بار را انجام دهد. این نوع زمانبندها بر اساس فاکتورهای مختلفی مثل طول صف، نرخ ورود کارها به صف و یا ترکیبی از این دو پارامتر عمل می کنند. تعیین این

[1] UnBalance

مساله را می توان بنا به محل قرار گرفتن زمانبند نسبت به صف پردازشها بیان کرد. بنابراین تقسیم بندی، سه نوع زمانبندی متفاوت به صورت زیر می توانیم داشته باشیم:

روشهای تنظیم طول صف(QAP)[۱]

در این روش زمانبند بین صف کارها و پردازشگر قرار می گیرد و عملیات تنظیم طول صف را با توجه به طول صفهای گره های مجاور انجام می دهد. در اینصورت همانطور که در شکل ۳-۶ دیده می شود اگر صف از حالت بالانس خارج شود، پردازشهای اضافی آن به گره های دیگر ارسال می شوند.



شکل۳-۶. روش مبتنی بر تنظیم طول صف (QAP) [۹۲].

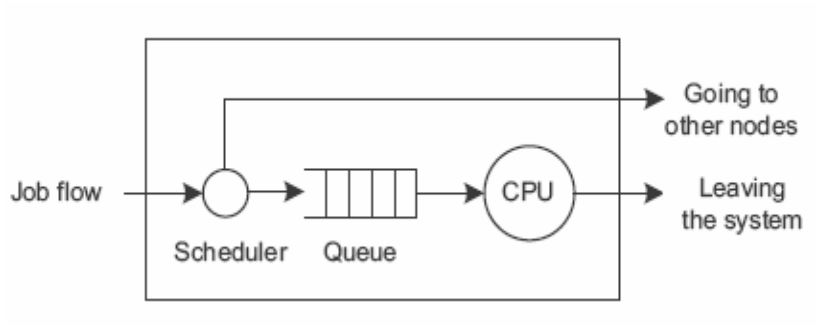
ELISA [۴۹] نمونه ای از الگوریتمهایی است که بر اساس تنظیم طول صف کار می کند. در این الگوریتم هر گره با توجه به اطلاعاتی که از همسایگانش دارد میانگین طول صف بر روی خودش و بر روی همسایه هایش را محاسبه می کند، گره هایی که انحراف آنها از میانگین کلی طول صف، به اندازه x کمتر باشد تشکیل "مجموعه فعال[۲]" را می دهند. سپس گره های پربار شروع به انتقال اضافه بار خود بر روی گره های مجموعه فعال می کنند. این عمل تا زمانیکه طول صف گره های پربار به حدود میانگین سیستم برسد، ادامه می یابد.

روشهای تنظیم نرخ ورود(RAP) [۳]

در این روش، زمانبند در ورودی صف قرار می گیرد. مطابق شکل ۳-۷ در هنگام ورود یک کار جدید به یک گره، زمانبند مشخص می کند که این کار باید همانجا پردازش شود یا اینکه باید به گره دیگری ارسال شود. با استفاده از این روش الگوریتمی به نام RLBVR[۴] [۹۲] ارائه شده است. جزئیات این الگوریتم در بخش بعدی خواهد آمد.

---

[۱] Queue Adjustment Policy
[۲] Active Set
[۳] Rate Adjustment Policy
[۴] Rate Load Balancing using Virtual Routing

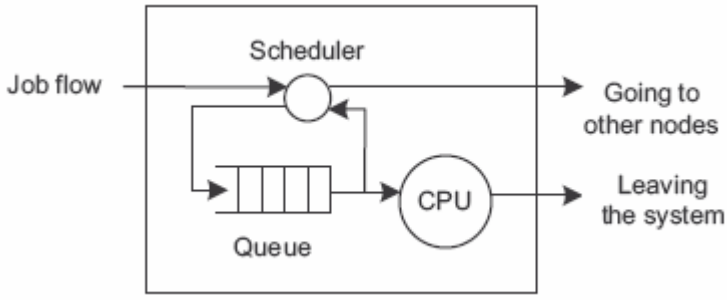


شکل۳-۷. روش مبتنی بر تنظیم نرخ ورود(RAP) [۹۲].

روش ترکیبی (QRAP)[1]

ترکیبی از روشهای تنظیم طول و نرخ ورود به صف است، یعنی زمانبند هم نرخ ورود کارها و هم طول صف را بصورتیکه در شکل۳-۸ نشان داده شده در هر گره کنترل و تنظیم می کند. استفاده از RAP در شرایط خاصی ممکن است نتواند به طور کامل موازنه بار را انجام دهد، در این شرایط ممکن است طول صف از یک حد آستانه ای بیشتر شود. در این حالت می توان از یک روش مبتنی بر طول صف، مثل ELISA استفاده کرد.

به عبارت دیگر می توان موازنه بار را در دو سطح، تنظیم نرخ ورود (RAP) به عنوان سطح اول و تنظیم طول صف (QAP) به عنوان سطح دوم انجام داد. QLBVR [۹۲] تنها الگوریتمی است که بر مبنای این رویکرد عمل می کند. البته QLBVR هم در واقع ترکیب دو الگوریتم ELISA و RLBVR می باشد که در دو سطح مذکور عمل می کنند. در ادامه به شرح نحوه کار RLBVR خواهیم پرداخت.



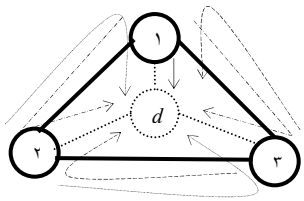
شکل ۳-۸ رویکرد ترکیبی(QLBVR) [۹۲].

روش کار الگوریتم RLBVR

همانطور که گفته شد این الگوریتم جزء خانواده الگوریتمهای تنظیم نرخ ورود می باشد. در الگوریتم QLBVR هم در سطح اول از همین روش استفاده می شود. در واقع روش کار RLBVR برگرفته از یک الگوریتم پایه ای تری با عنوان LBVR [۹۳] است.

LBVR، با استفاده از تکنیک مسیریابی مجازی[1]، الگوریتم موازنه بار را تبدیل به یک مساله مسیریابی می کند. برای این منظور یک گره مجازی که به تمام گره ها متصل است در نظر گرفته می شود (گره d در شکل ۳-۹). هزینه رسیدن به این گره مجازی، هزینه اجرا بر روی همان گره فرض می شود و داریم:

[1] Hybrid Policy

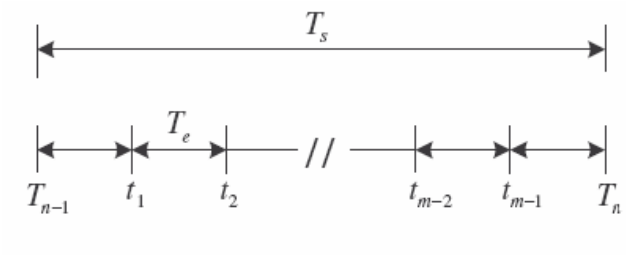


شکل ۳-۹. مسیر یابی مجازی.

$$(i,d) = \begin{cases} (i,d) & \text{فاصله هر گره } i \text{ تا گره} \\ (i,j) \longrightarrow (j,d) & \text{فاصله با استفاده از گره میانی } j \end{cases} \qquad (۸-۳)$$

در واقع تاخیر گرهی[۲] که همان تاخیر اجرای پردازش روی گره i است را به عنوان تاخیر ارتباطی[۳] (i,d) در نظر می گیریم. در این صورت هر کاری که به گره i با میانگین نرخ ورود λ_i وارد می شود از طریق هر یک از گره های دیگر $j(i<>j)$ به سمت d مسیریابی می شود. در این شرایط مساله موازنه بار تبدیل به یک مساله مسیریابی می شود که به آن "مسیریابی مجازی"گفته می شود. هدف این روش مسیریابی حداقل کردن تاخیر ارتباطی رسیدن به گره d برای هر کار گره i می باشد. در اینصورت برای هر گره i، هزینه مسیر (i,d) و $(i,j) \rightarrow (j,d)$ در نظر گرفته می شود. با استفاده از یکسری روشهای ریاضی می توان میزان بهینه نرخ مسیریابی از هر مسیر گره i را محاسبه کرد و در نهایت مقدار X_{ij} را به عنوان نرخ انتقال بهینه کار و β_i را به عنوان نرخ پردازش کار بدست آورد.

الگوریتم RLBVR در واقع یک مدل پویا و بر اساس LBVR می باشد. در این سیستم هر گره i بر نرخ ورود تقاضاها نظارت داشته و میزان متوسط λ_i را بر این اساس بدست می آورد. مطابق شکل۳-۱۰ این مقدار به عنوان نرخ ورود کار در بازه زمانی T_{n-1} و T_n می باشد. قبل از اینکه گره i شروع به محاسبه X_{ij} و β_i کند، یک پیغام مبتنی بر دریافت مقادیر X_{ij} و β_i به آنها می فرستد. همسایه ها هم در جواب مقادیر X_{ji} و β_j را به عنوان جواب به i ارسال می کنند، پس از این مرحله گره i شروع به انجام محاسبات می کند و در نهایت نیز یکسری از کارهای خود را به سمت همسایگان خود ارسال می کند. تبادل اطلاعات در بازه های زمانی T_{n-1} و T_n انجام می شوند، برای کم کردن حجم تبادلات (زیاد کردن بازه زمانی تبادلات بدون از دست دادن دقت) می توان بازه زمانی را به چندین زیر بازه تقسیم کرد، در این زیر بازه ها براساس اطلاعات دریافتی در T_{n-1}، می توان میزان بار فعلی گره فرستنده را حدس زد. در شکل۳-۱۰، T_s نشان دهنده بازه زمانی تبادل بار و T_e نشان دهنده بازه زمانی که محاسبه بار گره فرستنده در آنها انجام می شود است.



شکل۳-۱۰. بازه های زمانی تبادل و اندازه گیری بار [۹۲].

[۱] Virtual Routing
[۲] Nodal Delay
[۳] Communication Delay

نتیجه گیری:

الگوریتمهای موازنه باری که در درون زمانبند جای داده شده اند بار کمی را به سیستم تحمیل می کنند. چون در تمام گره ها زمانبند وجود دارد، بنابراین در تمام گره ها الگوریتم موازنه بار نیز اجرا می شود و از این لحاظ می توان گفت که این دسته از روشهای موازنه بار عادلانه عمل می کنند. ویژگی منحصر به فردی که بویژه در الگوریتمهای موازنه بار "مسیریابی مجازی" وجود دارد، در نظر گرفتن هزینه ارسال بار به گره مقصد است که در کمتر الگوریتم دیگری وجود دارد. ویژگی دیگر این است که از مفهوم دقیقتری برای سنجش بار استفاده شده است. در واقع مفهوم بار به صورت ترکیبی از طول صف و نرخ ورود بار به صف در نظر گرفته شده است. ایده جالب دیگر نحوه کم کردن حجم تبادلات است. با محاسبه بار گره فرستنده، در مقصد، الگوریتم فوق توانسته بازه زمانی تبادلات و در نتیجه حجم سربار لازم برای تبادلات را تا حد زیادی کاهش دهد. مشکلی که در این روش و به ویژه در RLBVR وجود دارد، حجم زیاد محاسبات در هر گره برای یافتن راه رسیدن به نرخ ورود متعادل است. همچنین الگوریتم قیاس پذیر نیست و نمی توان سربار مربوط به محاسبه بهترین مسیر مجازی را به تعداد زیادی گره تعمیم داد.

بهتر است برای استفاده از مزایای روش و در عین حال مصون ماندن از سربار زیاد، الگوریتم را در سطح تعداد محدودی گره اجرا کنیم. البته در این صورت ممکن است الگوریتم نتواند به طور کامل همگرا شود ولی تا حد زیادی تفاوت بار را در سیستم کاهش می دهد.

۳-٤- جمع بندی

موازنه بار عبارت است از توزیع یکسان بار بر روی منابع و حداکثر کردن کارایی آنها، ضمن کم کردن زمان اجرای کلی کارها. به طور کلی فرآیند موازنه بار دارای چهار روال اصلی است که عبارتند از: الف) روال اندازه گیری بار ب) روال تبادل اطلاعات ج) روال راه اندازی و د) روال انجام عملیات نهایی موازنه بار.

باید توجه داشت که گرید به لحاظ شرایط محیطی و ساختاری، تفاوتهای عمده ای با سیستمهای توزیع شده دارد، در نتیجه نمی توان از الگوریتمهای موازنه بار سیستمهای توزیع شده در گریدهای محاسباتی استفاده کرد.

در واقع مساله موازنه بار نوعی مسئله بهینه سازی است که در آن تابع هدف کم کردن میانگین زمان پاسخ کلی سیستم و یا افزایش بهره وری منابع سیستم است. مسئله موازنه بار همچون بسیاری از مسائل بهینه سازی جزء مسائل رام نشدنی است. بنابراین باید به دنبال جوابهای نسبتاً خوب بود. برای رسیدن به جوابهای نسبتاً خوب در یک زمان قابل قبول، بهتر است از روشهای اکتشافی استفاده کنیم.

یک روش اکتشافی کارا در موازنه بار، استفاده از روشهای الگوریتم ژنتیک است. یک الگوریتم ژنتیک برای موازنه بار در درون هر گره، در ARMS پیاده سازی شده است. این روش در سطح زمانبند می باشد، یعنی برای اجرای آن احتیاج به تحمیل پردازش اضافه ای به سیستم مدیریت منابع نیست و خود زمانبند کار موازنه بار را نیز انجام می دهد. به علت عدم قیاس پذیری الگوریتمهای ژنتیک این روش قابل تعمیم به کل گرید نبوده و تنها می تواند در سطح هر گره و برای موازنه بار درون آن، اجرا شود.

یکی از روشهای کارامد دیگر در حل مساله موازنه بار، استفاده از ایده زندگی مورچه هاست. در این فصل دو الگوریتم با استفاده از این ایده بیان شد که یکی از آنها برروی بستر ARMS بود. این الگوریتم موازنه بار را به صورت سراسری (بین گره ها) انجام می داد. اما این الگوریتم بسیار ابتدایی و ناکارامد بود.

در مرحله سوم بررسی، رویکرد مربوط به روشهای مبتنی بر زمانبند بیان شد. الگوریتمهای این دسته بر اساس محل قرارگرفتن زمانبند به سه دسته QAP و RAP و QRAP تقسیم بندی می شوند. ELISA به عنوان نمونه ای از الگوریتمهای QAP، RLBVR نمونه ای از الگوریتمهای RAP و QLBVR نمونه ای از الگوریتمهای QRAP می باشند که در این فصل مورد بررسی قرار گرفتند. مزیت این روشها که همگی در سیستمهای توزیع شده مطرح شده اند، سطح زمانبند بودن آنهاست که باعث می شود بار کمی به سیستم تحمیل شود. ضعف این روشها هم عدم قیاس پذیری در مقیاس وسیع است.

در نگاهی جامعتر به الگوریتمهای موازنه بار در سیستمهای توزیع شده و گرید، به این نتیجه می رسیم که ضعف عمده آنها نبود مفهوم واحد و جامعی برای سنجش بار در هر گره است. قابل اتکاء نبودن و بی اعتبار شدن سریع داده های مربوط به بار سایر گره ها، مشکلی است که در محیط گرید - به علت پویایی فوق العاده آن- عملیات موازنه بار را مشکل می سازد. مساله دیگر، میزان سربار ارتباطی و محاسباتی الگوریتم موازنه بار است. قیاس پذیری، پایداری و همگرایی از دیگر نیازهای الگوریتم موازنه بار می باشند.

در فصل بعد به بیان اهداف و ویژگیهای موازنه بار در محیط گرید محاسباتیARMS پرداخته و با توجه به ویژگیهای این محیط، سعی در ارائه یک راهکار موازنه بار ایده ال در این محیط می کنیم.

فصل ٤

MLBM مکانیزم موازنه بار چندسطحی در محیط محاسباتی گرید، با سیستم مدیریت منابع مبتنی بر عاملARMS

٤-١- مقدمه

پس از به وجود آمدن شبکه های کامپیوتری بستر مناسبی برای ارتباطات و همکاریهای علمی، تجاری، اجتماعی در سراسر جهان به وجود آمد. امروزه تخصصی شدن بیش از حد تمامی علوم، نیاز به تبادل اطلاعات و انجام کار دسته جمعی برای حل مسائل بزرگ را بیش از پیش کرده است. در همین راستا دانشمندان، پژوهشگران و حتی مراکز صنعتی بزرگ، تیمهای بزرگ تحقیقاتی را که ممکن است بیشتر اعضای آن در یک مکان متمرکز نباشند، تشکیل داده و از طریق محیط های همکاری[١] که امروزه با استفاده از فناوریهای جدید فراهم شده است، با هم همکاری می کنند[٧٨]. اما برای حل مسائل پیچیده امروزی علاوه بر همکاری اطلاعاتی، اغلب نیاز به منابع محاسباتی فوق العاده پرقدرتی است که در مقیاس وسیع و به راحتی در دسترس نیستند. این مشکل به ویژه برای گروه های همکاری فوق الذکر که اعضای آن در سراسر جهان با هم همکاری می کنند جدی تر است. برای حل این مشکل بسیاری از ابرکامپیوترها امکان دسترسی از راه دور[٢] به منابع موجود خود را فراهم می آورند. این رویکرد اگرچه مسائل قبلی از جمله عدم امکان دسترسی به منابع پرقدرت محاسباتی همچون ابر کامپیوترها را حل می کند، اما راه حل جامعی نبوده و مشکلات دیگری از جمله عدم امکان دسترسی همزمان به چند منبع که هر یک در مکان خاصی قرار دارند، عدم تضمین کیفیت سرویس، ایجاد گلوگاه محاسباتی و هزینه های اقتصادی بالا را بوجود می آورد[٧٩].

راهکاری که در چند سال اخیر برای دسترسی به منابع محاسباتی اتخاذ شده، به اشتراک گذاری منابع موجود در بین افراد علاقه مند می باشد. به این ترتیب افرادی که مایل هستند می توانند منابع محاسباتی ماشین های خود را در اختیار سایر افراد قرار داده و یا از منابع سایر ماشین ها استفاده کنند. اینگونه به اشتراک گذاری منابع بسیار فراتر از مدل های مشابه مانند محاسبات نظیر به نظیر[٣] می باشد، زیرا در آن مدلها تنها یک مورد از به اشتراک گذاری منابع، مانند اشتراک فایل[٤] مد نظر قرار می گیرد [٦٨]. اینگونه محیط های همکاری برای به اشتراک گذاری منابع اصطلاحاً "محیط های محاسباتی فراگیر[٥] " نامیده می‌شود [٩٤]. یک مدل آشنا از محیط های محاسباتی فراگیر، **گرید** [٢٠] می باشد، که می تواند در راستای اهداف متفاوت تشکیل شده و برای کاربردهای گوناگون استفاده شود.

گستردگی و تنوع منابع از سویی و حجم زیاد تقاضاهای کاربران از سویی دیگر، وجود یک سیستم موازنه بار بر روی منابع را برای ارائه سرویس مناسب به کاربران در محیط گرید ضروری می سازد. این مساله از لحاظ کم شدن زمان پاسخ تقاضاها، افزایش بهره وری منابع و همچنین مسائل اقتصادی مطرح در گرید، دارای اهمیت ویژه ای است.

در بخش قبل اصول و مبانی موازنه بار در سیستمهای توزیع شده و گرید، به ویژه روشهای ارائه شده در بستر مدیریتی ARMS مورد تحلیل و بررسی قرار گرفتند. با توجه به نتایج بدست آمده از این تحلیلها، در این فصل برآنیم که راهکار جدیدی برای موازنه بار در محیط ARMS که یک سیستم مدیریتی مبتنی بر عامل است ارائه دهیم. اما برای بیان بهتر موضوع، ابتدا به تعریف مسأله مورد بررسی پرداخته، مشکلات و محدودیت های موجود در آن را بررسی می کنیم، سپس مدلهای موجود در رابطه با مسأله مورد بحث را مرور کرده و دیدگاه خود را در حل مسأله بیان می کنیم. در نهایت سه الگوریتم برای موازنه بار بر روی منابع پیشنهاد می کنیم. پس از آن با تشریح ساختار محیط شبیه سازی به

---

[١] Collaborative Environments
[٢] Remote access
[٣] Peer to Peer
[٤] File Sharing
[٥] Pervasive Computing Environment

معرفی ابزارهای مورد استفاده در فرایند شبیه سازی می‌پردازیم. در انتها نیز با توصیف آزمایش های انجام شده، نسبت به ارزیابی عملکرد مدلها و اثبات صحت کارایی شبیه سازیها اقدام خواهیم کرد.

٤-۲- تعریف مساله

محیط گرید برای به اشتراک گذاری تعداد زیادی از منابع متنوع در بین کاربران گوناگون بکار می رود. هر کاربر، که ممکن است یک سازمان، دانشگاه و یا حتی یک فرد حقیقی باشد، می‌تواند به گرید پیوسته، نسبت به ارائه منابع خود به سایر کاربران اقدام نموده و یا در برنامه های خود از منابع آنها استفاده کند. برنامه هایی که از جانب کاربران به گرید تحویل داده می شوند، عموماً دو نوع رفتار از خود نشان می دهند. گروه اول خواص پردازشهای دسته ای[۱] را نشان می دهند. در این نوع تقاضاها کاربر معمولاً به علت پیچیدگی برنامه و عدم توانایی ماشین خود در اجرای برنامه، متقاضی اجرای برنامه بر روی گرید است. این دسته از کاربران معمولاً عجله ای برای اجرای برنامه خود ندارند. دسته دوم تقاضاها، رفتار برنامه های محاوره ای [۲] و بلادرنگ[۳] را از خود بروز می دهند. در این نوع تقاضاها معمولاً هدف کاربر از تحویل برنامه به گرید، اجرای سریعتر برنامه است. یک سیستم مدیریت منابع قابل انعطاف، بایستی با هر یک از این نوع تقاضاها برخورد مقتضی را به صورتی انجام دهد که زمان پاسخ برنامه ها (به خصوص در حالت محاوره ای و بلادرنگ) کاهش یافته و بهره وری منابع (به خصوص در حالت دسته ای) افزایش یابد. برای نیل به این هدف سیستم مدیریت منابع باید مجهز به روالهای موازنه بار باشد. انجام عملیات موازنه بار معمولاً بر روی تقاضاهای بلادرنگ، با توجه به حساسیت آنها به موعد زمانی[٤] و همچنین وقت گیر بودن روال موازنه بار- به جز در موارد بحرانی که ممکن است تقاضاها در مهلت زمانیشان اجرا نشوند- توصیه نمی شود. متقابلاً، شرایطی که تقاضاهای دسته ای دارند، موقعیت مناسبی برای موازنه بار است. در این شرایط بهره وری منابع افزایش و زمان پاسخ کلی برنامه ها احتمالاً کاهش خواهد یافت. اما قبل از آن به مرور مدل مدیریتیARMS که روالهای موازنه بار این پایان نامه برای آن طراحی شده اند می پردازیم.

ARMS یک مدل مدیریت منابع گرید است که عملکرد آن مبتنی بر تعامل عاملهای همکار و همگون می باشد. این سیستم مدیریت منابع در ابتدا فقط اجرای برنامه های بلادرنگ را پشتیبانی می کرد. در مراحل بعدی، با تقویت روال زمانبندی آن [۳۹]، قابلیت اجرای برنامه های دسته ای را نیز پیدا کرد. سپس یک مدل موازنه بار الگو گرفته از عملکرد گروهی مورچه ها برای موازنه برنامه های دسته ای بر روی آن ارائه شد. روش ارائه شده ابتدایی، ناعادلانه، قیاس ناپذیر بوده و از ظرفیتها و فرصت های موجود بهره خوبی نمی برد، نقص دیگر این روش عدم شفافیت برای کاربر بود که با اهداف سیستم مدیریت منابع در تناقض است. توضیحات بیشتر هر یک از این موارد در فصل ۳ بیان شده است.

با توجه به نقائص ذکر شده و با توجه به آنچه در فصل قبل (بخش ۳-۲-۱) در مورد ویژگیهای سیستم موازنه بار بیان شد، و اشاره به این نکته که این ویژگیها بعضاً متضاد بوده و ارضاء همه آنها در قالب یک الگوریتم واحد امکان پذیر نیست، در این پایان نامه قصد داریم با توجه به ویژگیهای منحصر به فرد سیستم مدیریت منابع ARMS، مدلی جامع برای موازنه بار گرید در آن ارائه دهیم که تا حد امکان ویژگیهای بیان شده را بپوشاند.

برای این منظور، مدل چند لایه ای ارائه داده ایم که در هر یک از لایه ها ی آن ویژگیهای خاصی از مجموعه ویژگیهای بیان شده برآورده می شود. در ضمن الگوریتمهای لایه های مختلف به هم کمک کرده و بعضی از ضعفهای یکدیگر را

---

[۱] Batch
[۲] Interactive
[۳] Real Time
[٤] Dead Line

نیز می پوشانند. در بخشهای بعدی ضمن بیان جزئی تر مدل ارائه شده، نشان داده می شود که این روش ویژگیهایی همچون عادلانه بودن، تطبیق پذیری، شفافیت از دید کاربر، سربار کم محاسباتی و ارتباطی، در نظر گرفتن سودمندی[۱] ارسال بار، سراسری بودن[۲] و خود سازماندهی را برآورده می کند. همچنین روش ارائه شده می تواند تا حد امکان مزایای روشهای موازنه بار سطح زمانبند و روشهای مستقل _ که در بخش ۳-۳-۱ شرح داده شده اند_ را نیز ارائه دهد.

٤-۳- مکانیزم ارائه شده

روش مبتنی بر گروه مورچه های ارائه شده با وجود تمام کاستیها، توانسته تا حد زیادی وضعیت عدم موازنه بار اولیه سیستم را بهبود بخشیده و بار گره ها را به حالت متوازن نزدیک کند. با بهینه سازی این الگوریتم می توان ضعفهای آنرا جبران کرده و در عین حال از ویژگیهای مثبت روشهای مبتنی بر گروه مورچه ها، همچون خود سازماندهی به نحو بهتری سود جست. البته بعضی از این ضعفها به ماهیت رفتار گروهی مورچه ها بر می گردد. برای مثال ناعادلانه بودن، که در الگوریتم اولیه [۳۸] نیز وجود دارد، ناشی از حرکت تصادفی مورچه ها بین گره ها (عاملها) است. در اینصورت بعضی گره ها ممکن است به کـرّات مورد موازنه بار قرار گیرند، در حالی که بعضی دیگر ممکن است برای مدت زیادی توسط مورچه ها ملاقات نشده و بارشان موازنه نشود.

یک راه حل برای رفع اینگونه نواقص، ترکیب ایده استفاده از مورچه ها با روشهای دیگری است که تا حدی عادلانه بودن و قیاس پذیری را در عین شفافیت و پایداری، برآورده سازد. در ضمن چون روش مورچه ها جزء روشهای مستقل می باشد، بهتر است روشی که قرار است با آن همراه شود در سطح زمانبند باشد سربار محاسباتی کمی را نیز برای سیستم به همراه بیاورد.

برای این منظور می توان از ایده مطرح شده در الگوریتم QLBVR [۹۲] استفاده کرد. این الگوریتم همانطور که در فصل قبل اشاره شد، الگوریتمی کارا و در سطح زمانبند است. ویژگی مهم این الگوریتم محاسبه سودمندی در هنگام موازنه بار است. به این معنی که این الگوریتم صرفاً به اینکه بار موازنه شود اکتفا نمی کند، بلکه به این نکته مهمتر نیز توجه می کند که اجرای برنامه روی همین گره بیشتر زمان می برد یا ارسال آن به سایر گره ها و اجرا در آنها. در واقع در این فرایند زمان مربوط به ارسال (تاخیر ارتباطی) نیز به عنوان یک پارامتر تصمیم گیری مطرح شده است. این پارامتر علی رغم اهمیت ویژه اش کمتر در سایر مدلها مورد بررسی قرار گرفته است و این یکی از دلایل مهم انتخاب این الگوریتم به عنوان جزئی از ساختار کلی مدل ارائه شده می باشد. اما همانطور که در فصل قبل هم اشاره شد، این روش در مقیاس تعداد گره های زیاد، سربار زیادی را به سیستم تحمیل می کند. به ویژه چون این روش در سطح زمانبند است، سربار محاسباتی زیاد، اشکال بزرگی برای آن به محسوب می شود. البته این الگوریتم (QLBVR) در سیستمهای توزیع شده که تعداد گره های آن محدود است، بکار گرفته شده است ولی نمی توان آن را به علت سربار محاسباتی زیاد، مستقیماً در سطح کل گرید بکار گرفت. دلیل سربار محاسباتی زیاد، همانطور که در قسمت ۳-۳ شرح داده شد، محاسبات سنگین برای بدست آوردن نرخ ورود بهینه هر گره و همچنین زمانبر بودن تکنیک مسیریابی مجازی برای تعداد گره های زیاد است. در مدل ارائه شده این پایان نامه، سعی کرده ایم مدل تغییر یافته ای از این الگوریتم ارائه دهیم که در آن هم بار محاسباتی کمی به سیستم تحمیل شود و هم هزینه مسیریابی مجازی حداقل باشد.

---

[۱] Profitability
[۲] Global

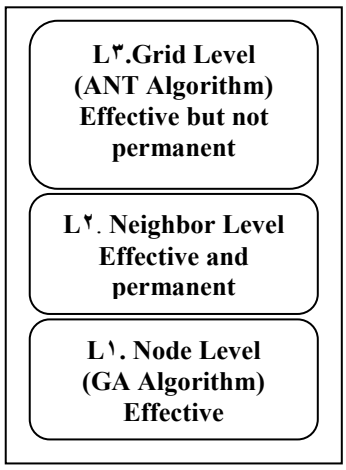
استفاده از این الگوریتمها هیچ منافاتی با استفاده از الگوریتم ارائه شده در [۴۰] که در سطح گره می باشد و با یک زمانبندی مناسب سعی در استفاده بهینه از پردازنده (های) گره ها می کند، ندارد. (این الگوریتم در بخش ۳-۳-۲ از فصل ۳ به طور کامل شرح داده شده است).

با استفاده از روش مبتنی بر استفاده از الگوریتم ژنتیک [۴۰] برای بهینه سازی زمانبندی در سطح هر گره، الگوریتم بهینه شده مورچه ها به عنوان یک روش مستقل در سطح بین گره ها (در سطح کل گرید) و یک الگوریتم موازنه بار دیگر که " **در سطح زمانبند و به طور دائم اجرا شود ولی موازنه بار را در سطح گره های همسایه، با توجه به هزینه ارسال بار انجام دهد و همچنین بار کمی را به سیستم تحمیل کند**" می توان روش چند سطحی جدیدی ارائه داد. این سطوح می توانند در کنار هم پردازشها را در سطح گرید به نحو خوب و کاملی، با میزان سربار کم موازنه کنند. ما این مدل جدید را MLBM نام نهاده ایم. لازم به ذکر است در سیستم موازنه بار MLBM از مفهوم دقیقتری نیز برای بیان میزان بار گره ها استفاده می شود که در قسمتهای بعد توضیح داده می شود.

در یک جمع بندی کلی می توان مدل MLBM را به صورت زیر بیان کرد:

- در سطح اول (سطح محلی) از الگوریتم زمانبندی مبتنی بر الگوریتم ژنتیک [۴۰] استفاده می شود. در این مرحله یک ترتیب بهینه برای اجرای پردازشهای در حال انتظار در هر گره مشخص می شود.
- سطح دوم (سطح همسایگی)، الگوریتم جدیدی با الهام از الگوریتم QLBVR است. این الگوریتم که در سطح زمانبند اجرا شده و در سطح همسایه های هر گره عمل می کند، می تواند در در همان ابتدای شروع به کار پردازش عملیات موازنه بار را تا حد زیادی انجام دهد. این روش عملیات خود را با توجه به هزینه ارسال (سودمندی ارسال) انجام می دهد و از این لحاظ که بار زیادی به سیستم تحمیل نمی کند، نسبت به الگوریتم اولیه (QLBVR) برتری دارد.
- در سطح سوم (سطح گرید)، می توان از الگوریتم بهینه شده مورچه ها استفاده کرد تا موازنه بارهای لازم را انجام دهد. در این سطح بارها بین گره های مختلف (به طور سراسری) پخش می شود. این الگوریتم به نسبت روش اولیه [۳۸] بار کمتری را به سیستم تحمیل می کند. استفاده از لایه دوم نیز موجب کم شدن تعداد مورچه ها و در نتیجه کمتر شدن سربار الگوریتم مورچه ها می شود.

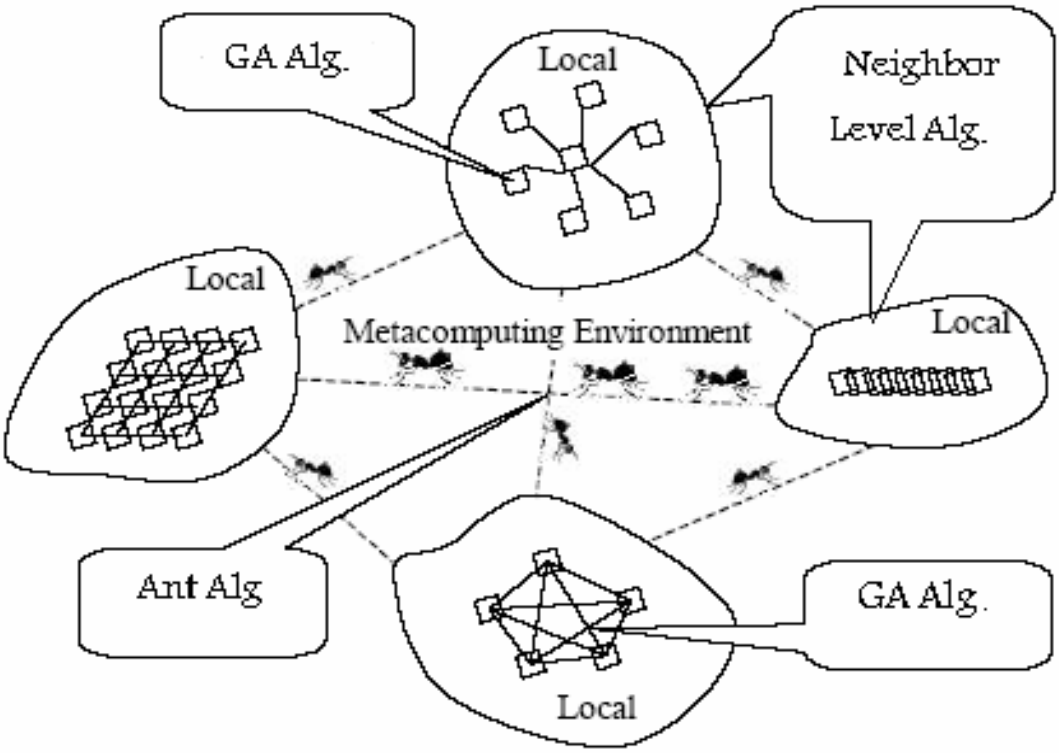
شکلهای ٤-١ و ٤-٢ باعث درک بهتر راهکار MLBM می شوند.



شکل٤-١. ساختار لایه ای راهکار ارائه شده(MLBM)

باید توجه داشت که راهکار استفاده از مورچه ها و راهکار سطح همسایگی، هر یک به تنهایی و مستقل از دیگری یک روش موازنه بار محسوب می شوند و تعامل آنها با یکدیگر باعث از بین رفتن معایبشان و عملکرد قویتر آنها می گردد. در واقع وجود سطح دوم (سطح همسایگی) باعث احتیاج به تعداد مورچه کمتری در مکانیزم مورچه ها خواهد شد، که این به نوعی افزایش کارایی کل سیستم محسوب می گردد. همچنین با استفاده از الگوریتم سطح همسایگی، هر گره برای خودش تصمیم می گیرد و منتظر رسیدن مورچه نمی ماند. بدین وسیله ضعف روش مورچه ها در سرکشی از مکانهایی که شانس کمتری برای بازدید دارند، برطرف گشته و وضعیت گره های نامتوازن در یک زمان قابل پیش بینی تا حد زیادی ترمیم می گردد. در طرف دیگر، الگوریتم سطح همسایگی هم به تنهایی دارای معایبی است که استفاده از مدل مورچه ها تا حد زیادی آنها را برطرف می سازد. یکی از این ایرادات، سرعت کم پخش بار در کل سیستم است. ایراد دیگر، نداشتن قدرت موازنه بار در ورای شعاع همسایگی هر گره است. به طور کلی می توان گفت که الگوریتم سطح همسایگی، یک روش پیش گیری از نامتوازن شدن سیستم است و الگوریتم مبتنی بر گروه مورچه ها، یک روش کشف و ترمیم عدم توازن بار در سیستم است.

به این ترتیب، منطق کلی مدل ارائه شده بدین صورت است که در مرحله اول با زمانبندی خوب و کارا از هر گره استفاده بهینه را ببریم [٤٠]. برای گره هایی که نسبت به همسایه هایشان اضافه بار دارند، با استفاده از روش سطح همسایگی و با در نظر داشتن هزینه مسیر، سعی در موازنه بار آنها به همسایه ها می کنیم. در اینصورت اضافه بار هر گره، در سطح یک ناحیه پخش می شود. همزمان با این الگوریتم، در سطح سوم الگوریتم بهینه شده مورچه ها، سعی در از بین بردن ناحیه های پربار و سراسری ساختن موازنه بار بین گره های مختلف در شعاع وسیعی، یعنی در سطح کل گرید می کند.



شکل٤-٢. سطح تاثیر گذاری الگوریتمهای سطوح مختلف MLBLM.

در قسمتهای بعد به جزئیات مربوط به راهکار ارائه شده در بکار گیری هوش گروهی مورچه ها، جزئیات مربوط به راهکار موازنه بار سطح همسایگی و در نهایت ترکیب این دو در قالب مدل MLBM خواهیم پرداخت. همه این راهکارها با توجه به سیستم مدیریت منابع ARMS و استفاده از ویژگیهای منحصر به فرد آن ارائه شده است.

٤-٣-١- راهکار موازنه بار با استفاده از اکوسیستم مورچه های هوشمند

همانطور که اشاره شد، محیط گرید، محیطی پویاست و هیچ مدیریت واحدی برای آن وجود ندارد. میان افزارهای[١] این محیط باید به گونه ای عمل کنند که این خلأ مدیریت جبران شود. به عبارت دیگر این میان افزارها باید تا حد امکان خودسازمانده باشند و بتوانند بدون هیچ عامل ناظری، هدف مورد نظر را برآورده سازند. براساس آنچه در فصل ٣ بیان شد، ایده استفاده از الگوی زندگی حشرات اجتماعی یکی از روشهای ایجاد خود سازماندهی است.

برای این منظور، در مکانیزم جدید، اکوسیستمی[٢] از مورچه های هوشمند ارائه می دهیم که با استفاده از آنها بار به صورت یکنواختی در سراسر گرید پخش می شود. این مورچه های هوشمند دارای حافظه، قدرت یادگیری و انطباق پذیری هستند. تعامل این مورچه های هوشمند و خود مختار باعث موازنه بار در سطح گرید خواهد شد.

منظور از اکوسیستم این است که جمعیت مورچه ها و همچنین طول حیات آنها (تعداد گامها) به طور پویا و بنا به شرایط سیستم تغییر می کند. در این اکوسیستم مورچه ها بنا به نیاز و شرایط سیستم به وجود آمده، عملیات موازنه بار را در طول حیات خود و با توجه به شرایط محیطی انجام می دهند. در شرایط خاصی که مورچه احساس کند به وجود آن نیازی نیست، دست به خودکشی می زند و در شرایطی هم که مورچه احساس کند سیستم در شرایط نامتوازنی است، ممکن است مورچه های جدیدی متولد کند. به این ترتیب جمعیت مورچه ها و طول عمر آنها که دو فاکتور اساسی در کارایی این روش محسوب می شوند به طور پویا و بنا به شرایط سیستم تنظیم می گردند.

همچنین این مورچه ها نسبت به هر گرهی که در طی یک دوره زندگیشان (در طی گامهایشان) ملاقات می کنند حساسند و مشخصات آنها را برای تصمیم گیریهای آینده ثبت وضبط می کنند. علاوه براین هر مورچه در مکانیزم جدید به جای $2m+1$ گام (طبق آنچه در [٣٨] آمده)، m گام پرش می کند و در انتهای این m گام هم به جای بالانس تنها دو عنصر، سعی در موازنه بار k گره پربار با k گره کم بار می کند. این طرز عملکرد باعث همگرایی سریعتر سیستم با وجود تعداد مورچه های کمتر و همچنین سربار ارتباطی کمتر می شود. در ادامه، ضمن شرح جزئیات راهکار جدید، با فرموله سازی مساله، افزایش کارایی آن نسبت به روش قبلی را به صورت ریاضی نیز اثبات می کنیم. شبیه سازیهای انجام شده در قسمتهای بعدی همین فصل نیز این افزایش کارایی را تایید خواهند کرد.

چگونگی ایجاد مورچه های جدید

در ابتدا اگر گرهی متوجه شود که بیش از حد بار شده است، مورچه ای با تعداد گامهای کم ایجاد می کند تا بارش را به سرعت موازنه کند.

همانطور که در فصل قبل شرح داده شد، نحوه محاسبه بار گره ها، تاثیر زیادی در جلوگیری از تصمیم گیریهای اشتباه یا به عبارت دیگر پایداری سیستم دارد. راههای زیادی وجود دارد که یک گره می تواند بار خودش را اندازه گیری کند. در این میان، فاکتورهایی همچون طول صف کارهای منتظر، نرخ ورود کارها به گره و میانگین زمان پاسخ، بیشتر مورد استفاده بوده اند. طبق آنچه در [٥٥] بیان شده است، ترکیبی از فاکتورهای فوق می تواند پارامتر مناسبی برای اندازه

[١] Middleware
[٢] Echo System

گیری بار گره ها باشد. اما این معیار ترکیبی هم خیلی دقیق نیست. در این پایان نامه مدل دقیقتری که در آن میزان بار هر گره بر اساس میزان زمان لازم برای اجرای پردازشهای فعلی، نرخ خروج کار از گره و نرخ ورود کار به آن، تعیین می شود، ارائه شده است. رابطه ٤-١٧ بیان کننده چگونگی محاسبه بار در روش جدید می باشد.

حرکت و تصمیم گیری

یک مورچه پس از تولد شروع به جمع آوری اطلاعات گره های اطراف می کند. در هر گام مورچه یکی از گره های همسایه را به طور تصادفی انتخاب کرده و به آن پرش می کند. یک مورچه پس از تولد ممکن است چندین بار عمل موازنه بار را برای گره هایی که اطلاعات آنها را جمع آوری کرده انجام دهد. پس از هر بار انجام عمل موازنه بار، مورچه دوره جدیدی را آغاز می کند.

همچنین به هر مورچه فضای حافظه ای تخصیص داده شده است. مورچه با استفاده از این فضای حافظه مشخصات محیط (مشخصات گره هایی که ملاقات می کند) را در طول حرکتش نگهداری می کند. این فضای حافظه به دو قسمت تقسیم می شود. در یک قسمت لیست عناصر کم بار و در قسمت دیگر لیست عناصر پرباری که در طول دوره فعلی زندگی مورچه انتخاب شده اند نگهداری می شود.

الگوریتم بررسی دور[1]

یکی از نقائص موجود در مدل ارائه شده اولیه [٣٨] این بود که امکان داشت شرایطی به وجود بیاید که در آن مورچه بارها دو یا چند گره را ملاقات کنند. این مساله به این دلیل رخ می دهد که مورچه هیچ هوشمندی و کنترلی نسبت به مکانهایی که ملاقات کرده نداشته و نمی توانست آنها را به خاطر بسپارد. شرایط بدتر زمانی روی می داد که دو یا چند مورچه، همزمان یک گره را به عنوان مقصد موازنه بار (مکانی که اضافه بار را به آن می فرستند) انتخاب کنند. پر واضح است که در چنین شرایطی گره مقصد، خیلی سریع اشباع می شود و هر آنچه پس از آن از سایر فرستنده ها دریافت کند، باعث پربار شدن آن می شود. به عبارت دیگر چنین شرایطی منجر به اتخاذ تصمیمات نادرست (عدم پایداری) می گردد؛ چرا که الگوریتم، یک گره کم بار سیستم را، تبدیل به یک گره پربار کرده است. همانطور که در [٦٧] اشاره شده است، دو راه برای جلوگیری از این مساله وجود دارد.

روش اول استفاده از اسید مورچه است. ما از این روش برای جلوگیری از انتخاب یک گره، توسط بیش از یک مورچه استفاده کرده ایم. در اینصورت هر بار که مورچه ای گرهی را به عنوان پربار یا کم بار انتخاب می کند، اسید خود را در آن گره به جای می گذارد. این اثر سبب می شود که این گره توسط سایر مورچه های عبوری از آن گره انتخاب نشود. به عبارت دیگر از اسید به جای مانده از مورچه به عنوان یک روش قفل گذاری استفاده کرده ایم.

روش دوم استفاده از روالهای بررسی دور[2] است. برای استفاده از این روال بایستی به مورچه حافظه داده شود. از این حافظه اعطا شده به مورچه در قسمتهای بعدی نیز استفاده می شود. با استفاده از این روش، هر بار که یک مورچه به یک گره وارد می شود، ابتدا حافظه اش را چک می کند تا مشخص شود که آیا این گره تا کنون ملاقات شده است یا خیر. این کار برای جلوگیری از به وجود آمدن دور در حرکت مورچه انجام می شود. در این شرایط اگر گره تا کنون ملاقات نشده باشد، مورچه می تواند در مورد وضعیت آن گره تصمیم گیری کند (یعنی مشخص کند که کم بار، پر بار یا متوازن است).

---

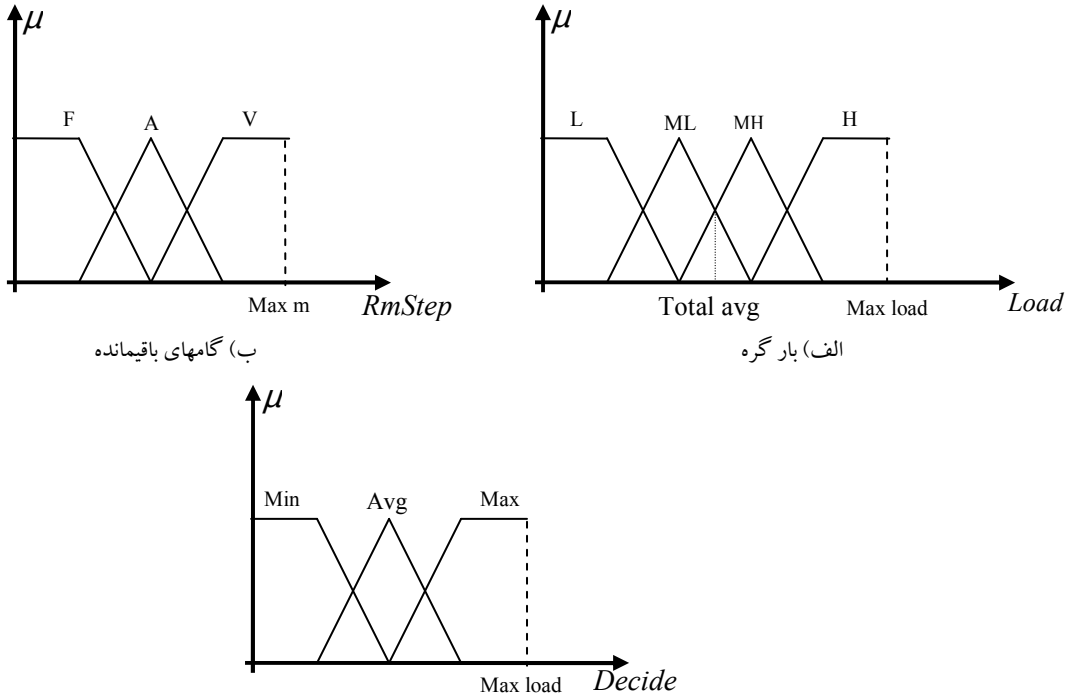
[1] Cycle Checking
[2] Cycle

الگوریتم تصمیم گیری

همانطور که گفته شد، مورچه پس از ورود به یک گره، اگر آن گره تا کنون ملاقات نشده باشد، در مورد وضعیت آن گره تصمیم گیری می کند (یعنی مشخص کند که کم بار، پر بار یا متوازن است). این تصمیم گیری بر اساس دانش داخلی مورچه که از طریق محیط بدست آمده است، انجام می شود.

مقدار بار یک گره یک متغیر زبانی[1] می باشد، یعنی می توان بار گره را به صورت خیلی کم بار،کم بار، پربار، خیلی پربار، موازنه بیان نمود. علاوه براین وضعیت بار هر گره (کم بار بودن، پربار بودن یا متوازن بودن) نسبت به شرایط سیستم یا به عبارتی بر اساس اطلاعاتی که تا کنون مورچه بدست آورده است، بیان می شود. از اینرو با توجه به قدرت منطق فازی در ایجاد نگاشت ریاضی متغیرهای زبانی و انطباق پذیری آن با شرایط مختلف، الگوریتم تصمیم گیری مورچه در مورد وضعیت گره به صورت تطبیقی و با به کارگیری منطق فازی انجام می شود. در این تصمیم گیری، مورچه میزان بار کاری فعلی گره و تعداد گامهای باقیمانده خود را به عنوان ورودی به سیستم فازی داده و سپس وضعیت بار گره (پربار، کم بار و متوازن) را با توجه به میزان حافظه باقیمانده خود مشخص می کند.

میانگین کل باری که تا کنون توسط مورچه مشاهده شده و در حافظه آن ثبت است، به عنوان دانش داخلی مورچه نسبت به محیط در نظر گرفته می شود. همانطور که در شکل ٤-٣-الف دیده می شود، مورچه از این دانش برای ساخت توابع عضویت[2] بار گره فعلی استفاده می کند. مطابق آنچه در شکلهای ٤-٣-ب و ج دیده می شود، توابع عضویت مربوط به تعداد گامهای باقیمانده و همچنین توابع عضویت خروجی سیستم فازی (تصمیم)، بر اساس یک حد آستانه[3] و به صورت تجربی در نظر گرفته شده است.



ب) گامهای باقیمانده

الف) بار گره

ج) خروجی (وضعیت گره)

شکل ٤-٣. توابع عضویت مجموعه های فازی .

[1] Linguistic Variable
[2] Membership Functions
[3] Threshold

این سیستم استنتاجی می تواند به صورت یک رابطه ریاضی و تابع متناظر آن نیز بیان شود:

$$R_A : Load < L, ML, MH, H > * RmStep < F, A, V > \rightarrow Decide < Min, Avg, Max >$$
(۱-٤)

$$f(x) = \frac{\sum_{r=1}^{12} \bar{y}^r * \mu_{A_1^r}(x_1) * \mu_{A_2^r}(x_2)}{\sum_{r=1}^{12} {}_{A_1^r}(x_1) * \mu_{A_2^r}(x_2)} \qquad (۲-٤)$$

که در آن r تعداد قوانین، $\bar{y}^r$ مرکز تابع عضویت در مجموعه فازی *Decide*، x_i بیان کننده مقدار ورودی و $\mu_{A_i^r}(x_i)$ میزان عضویت x_i در مجموعه فازی A_i در قانون r است. برای مثال اگر مورچه در گامهای آخر خود باشد و با گرهی که بار آن در مجموعه فازی ML است مواجه شود، آن را به صورت کم بار در نظر می گیرد تا بتواند حافظه خود را تا حد امکان اشباع کند. در این شرایط قانون زیر فعال[۱] می شود.

IF *Load=ML* AND *RemainStep=Few* THEN *DECIDE=Min*

این موتور استنتاج از فازی ساز یکنواخت[۲]، موتور استنتاج ضربی[۳] و غیر فازی ساز میانگین مراکز[٤] استفاده می کند.

به این ترتیب مورچه می تواند تصمیم مناسب را اتخاذ کند. بنابراین اگر نتیجه تصمیم گیری پربار یا کم بار بودن گره باشد، اطلاعات گره به لیست عناصر پربار *(Max-list)* یا کم بار *(Min-list)* مورچه اضافه می شود، متعاقبا شمارنده متناظرعناصر پربار، کم بار و یا متوازن ملاقات شده مورچه نیز یک واحد افزایش می یابد. این شمارنده ها نیز به نوعی بیانگر دانش مورچه نسبت به محیط اطراف هستند. کاربرد این دانش در قسمتهای بعدی شرح داده می شود.

موازنه بار در سطح مورچه ها

با دقت بیشتر برروی رفتار مورچه ها و تعاملات آنها، متوجه می شویم که وقتی دو مورچه با هم برخورد می کنند، برای لحظه ای توقف می کنند و شاخکهایشان را به هم می زنند _ آنها احتمالا این کار را برای تشخیص هم گروهیهایشان انجام می دهند _ ما از این مساله برای موازنه بار در سطح مورچه ها استفاده می کنیم. این اولین استفاده از این ایده در روشهای بهینه سازی با الگوبرداری از مورچه ها است.

به طور دقیقتر، با توجه به ساختار سیستم ارائه شده، ممکن است دو یا چند مورچه به طور تصادفی در یک گره هم را ملاقات کنند. همانطور که در بالا اشاره شد، هر کدام از این مورچه ها ممکن است مشخصات یکسری از گره های پربار و کم بار را جمع کرده باشند. تعداد این اطلاعات (مربوط به عناصر پربار و کم بار) در هر مورچه الزاماً مساوی نیست. مثلاً یکی ممکن است در حافظه خود اطلاعات ٤ عنصر پربار و ۲ عنصر کم بار را داشته باشد در حالیکه مورچه دیگر _ که در همان مکان قرار دارد _ دارای اطلاعات دو عنصر پربار و ٦ عنصر کم بار است. در این شرایط این دو مورچه می توانند بار خود را با مبادله اطلاعاتشان بالانس کنند. ما این عمل موازنه بار در سطح مورچه ها نامیده ایم. در مثال قبلی پس از موازنه بار در سطح مورچه ها، دو مورچه هم مکان دارای مشخصات ۳ گره پربار و ٤ گره کم بار در حافظه هایشان هستند. این مساله باعث کارایی بهتر در زمان موازنه بار خواهد شد. این عملیات قابل تعمیم به بیش از دو مورچه نیز می باشد.

---

[۱] Fire
[۲] Single Tone
[۳] Product
[٤] Center of Gravity

در واقع وقتی دو یا چند مورچه هم مکان، اطلاعاتشان را مبادله می کنند، شعاع حرکتیشان به دامنه بزرگتری توسعه می یابد که این منجر به ایجاد درک بهتری از محیط می شود. ایده مشابه ای وجود دارد که از اسید به جای مانده از مورچه ها در حال حرکت الهام گرفته شده است. سایر مورچه ها می توانند این مورچه را با استفاده از اسید برجای مانده دنبال کنند. این ایده در بسیاری از مسائل بهینه سازی با استفاده از دسته مورچه ها، از جمله *Messor* [۲] استفاده شده است. اما تفاوت ریزی بین این دو ایده وجود دارد. اطلاعات باقی مانده از مورچه در یک مکان ممکن است در طی زمان نامعتبر شود. برخی این مساله را با ایدهٔ تبخیر حل کرده اند، اما ایدهٔ تبخیر در بعضی شرایط نمی تواند کار ساز باشد، این مساله به ویژه در شرایطی که ما احتیاج به اطلاعات قابل اطمینان داریم مهم است. مثلا در گرید، که در آن اطلاعات بار به سرعت تغییر می کند استفاده از اطلاعات باقی مانده از سایر مورچه ها خیلی قابل اطمینان نیست و ممکن است منجربه تصمیم گیری اشتباه توسط سایر مورچه ها شود. اما در ایده جدید، دانش مبادله شده مورچه ها کاملا قابل اطمینان است.

ایجاد مورچه های جدید در حین حرکت

در شرایط خاص ــ به خصوص زمانیکه دوره زندگی مورچه طولانی است ــ در حالی که مورچه حرکتش را ادامه می دهد ممکن است حافظه اش پر شود ولی در ادامه حرکت خود همچنان به عناصر پربار یا کم بار برخورد کند. در این شرایط اگر گره فعلی که مورچه در آن است پربار باشد، مورچه ی در حال حرکت یک مورچه جدید با تعداد گامهای از پیش تعیین شده متولد می کند، و اگر این مورچه در حال حرکت با گرهی خیلی کم بار برخورد کند، اطلاعات آن گره را با بزرگترین عنصر موجود در لیست عناصر کم بار خود جانشین می کند.

این مساله باعث تطبیق پذیری جمعیت مورچه ها با شرایط محیط می گردد. در این صورت اگر تعداد عناصر پربار سیستم زیاد باشد، جمعیت مورچه ها به طور خودکار زیاد می شود.

موازنه بار و شروع یک دوره جدید

وقتی پرشهای مورچه تمام شد، مورچه عملیات بالانس بین عناصر پربار و کم باری که در طی حرکتش جمع آوری کرده را انجام می دهد و بار را بین تمام این گره ها به طور یکنواخت توزیع می کند. به علت استفاده از عملیات موازنه بار در سطح مورچه، تعداد عناصر در لیست عناصر پربار و کم بار به احتمال زیاد به هم نزدیکند و این باعث بهتر شدن کارایی می شود.

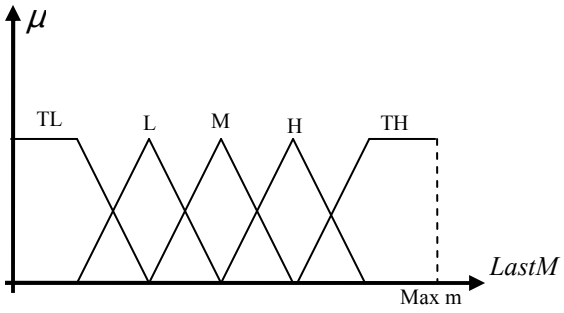
پس از موازنه بار، مورچه باید مجدداً برای آغاز یک دوره جدید آماده شود. یکی از فیلدهایی که برای آغاز دوره جدید باید مقدار دهی شود تعداد گامهای مورچه در دور بعدی (m) است. اما همانطور که قسمت قبل نیز گفته شد، تعداد این گامها باید متناسب با شرایط سیستم باشد. یعنی اگر مورچه به این نتیجه رسید که اختلاف بار شدیدی در سیستم وجود دارد، باید به نحوی عمل کند که فرکانس عملیات موازنه بار افزایش یافته و بار به سرعت در تمام سیستم پخش شود. بالعکس، در شرایطی که مورچه احساس کند سیستم تا حدی به حالت بالانس نزدیک است باید فرکانس انجام موازنه بار را کاهش دهد. به این ترتیب مورچه رفتار تطبیق پذیری در طول زندگی اش از خود نشان می دهد. تشخیص اینکه سیستم در حالت دور از تعادل و یا نزدیک به تعادل است باز هم می تواند با استفاده از دانش مورچه نسبت به محیط حاصل شود. این دانش می تواند با انجام پردازش بر روی اطلاعاتی که مورچه در دوره قبلی خود و دوره های قبلی خود از محیط بدست آورده انجام شود. اطلاعات دوره قبل عبارتست از تعداد گره های پربار، کم بار و متوازنی که در طی گام قبلی ملاقات شده است. اطلاعات مربوط به وضعیت محیط در دوره های قبلی نیز از تعداد گامها در مرحله قبل $(LastM)$ قابل استنتاج است. به عبارت دیگر زیاد بودن تعداد گامهای مورچه در مرحله قبل نشان دهنده این است که مورچه در دوره های قبلی وضعیت سیستم را نزدیک به حالت بالانس درک کرده و در نتیجه آن تصمیم به کم کردن فرکانس موازنه بار (طولانی کردن تعداد گامهای خود) گرفته است. بالعکس، تعداد گامهای کم در دوره

قبل، بیان کننده این است که مورچه در دوره های قبلی شرایط بار سیستم را دور از حالت تعادل درک کرده و در نتیجه تصمیم به افزایش فرکانس (کاهش تعداد گامهای خود) گرفته است.

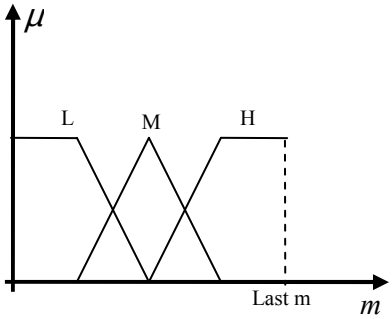
مسلماً این گونه تغییر رفتار در دوره های مختلف حیات مورچه نباید ناگهانی باشد. بلکه این تغییر رفتار بایستی به صورت تطبیق پذیر و با توجه به فاکتورهای موثر فوق الذکر صورت گیرد. به علت قدرت منطق فازی در ایجاد تطبیق بین پارامترهای مختلف یک مساله و با در نظر گرفتن تعداد گامهای مورچه به عنوان یک متغیر زبانی مثل کم، متوسط، طولانی، منطقی است برای تعیین تعداد گامهای مرحله بعد مورچه نیز از منطق فازی استفاده کنیم.

کنترل کننده فازی مورد نظر، تعداد گامهای دوره بعدی مورچه *(NextM)* را بر اساس تعداد گره های کم بار، پربار، متوسط و همچنین تعداد گامهای مرحله قبل مشخص می کند. همانطور که گفته شد تعداد گره های پربار، کم بار و متوسط که در طول مرحله قبل توسط مورچه ملاقات شده اند نشان دهنده وضعیت اخیر سیستم است، درحالیکه تعداد گامهای مورچه در مرحله قبل *(LastM)*، نشان دهنده تاریخچه زندگی مورچه است.

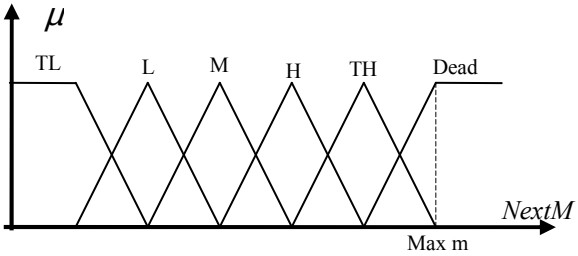
بنابراین می توانیم توابع عضویت مجموعه های فازی کنترل کننده مربوطه را به صورت آنچه در شکل ٤-٤ نشان داده شده است، بیان کنیم.



(الف) تعداد کل گامها در آخرین مرحله



(ب) تعداد گره های پربار، کم بار و متوسط ملاقات شده توسط مورچه درمرحله قبل



(ج) خروجی (تعداد گامها دوره بعدی)

شکل٤-٤. توابع عضویت مجموعه های فازی .

این کنترل کننده فازی را نیز می توان به صورت یک رابطه به صورت زیر تعریف کرد. همچنین تعداد گام مرحله بعد نیز توسط تابع ٤-٤ بیان می شود.

$$R_B : MaxCount\langle l,m,h\rangle * MinCount\langle l,m,h\rangle * AvgCount\langle l,m,h\rangle * LastM\langle TL,L,M,H,TH\rangle \rightarrow NextM\langle TL,L,M,H,TH,Dead\rangle \qquad (٣-٤)$$

$$f(x) = \frac{\sum_{r=1}^{135} \bar{y}^r * \prod_{i=1}^{4} \mu_{B_i^r}(x_i)}{\sum_{r=1}^{135} \prod_{i=1}^{4} \mu_{B_i^r}(x_i)} \qquad (٤-٤)$$

که در آن x_i نشان دهنده داده ورودی به سیستم، $\mu_{B_i^r}(x_i)$ نشان دهنده میزان عضویت ورودی x_i در مجموعه فازیB_i قانون r ام و $\bar{y}^r$ نیز مرکز مجموعه فازی بیان شده در قسمت آنگاه قانون r است. علاوه بر این همانطور که در رابطه ٤-٣ دیده می شود، در این سیستم استنتاج ٤ ورودی و١٣٥ قانون داریم.

در این سیستم تعداد زیاد گره های کم بار و به ویژه متوازن نشان دهنده این است که سیستم در حالت تقریباً متعادلی است و در نتیجه تعداد گام مورچه در دوره بعدی باید افزایش یابد. این نحوه عملکرد باعث کاهش فرکانس انجام عملیات موازنه بار می شود.

در حالت حدی فرض کنیم تعداد گامهای مورچه به سمت بی نهایت میل کند، در نتیجه فرکانس انجام موازنه بار کاهش یافته و تاثیر آن به سمت صفر میل می کند. بر این اساس می توان نتیجه گرفت که مورچه ای با تعداد گامهای زیاد تاثیری در بالانس سیستم ندارد ولی همچنان سربار ارتباطات خود را به سیستم تحمیل می کند. در این شرایط چون مورچه سودی برای سیستم ندارد باید از بین برود (مورچه باید خودکشی کند). این مرحله آخرین حلقه اکوسیستم ذکر شده می باشد.

بنابراین، اگر تعداد گامهای مشخص شده برای مرحله بعد*(NextM)*، طبق آنچه در شکل ٤-٤-ج نشان داده شده است، در قسمت*Dead* قرار بگیرد، مورچه دیگر دوره جدیدی را شروع نمی کند. پیگیری این اکوسیستم برای همه مورچه های سیستم، منجر به همگرایی نهایی روش ارائه شده می گردد.

٤-٣-٢- ارزیابی کارایی مکانیزم مبتنی بر گروه مورچه های هوشمند

در این قسمت برای نشان دادن کارایی روش ارائه شده، ابتدا چند پارامتر آماری را مورد بحث قرار می دهیم.

کارایی:

در این قسمت اثبات می کنیم که روش ارائه شده نسبت به روش اولیه [٣٨] کارایی بهتری را نتیجه می دهد.اما ابتدا چند معیار متعارف را در موازنه بار توضیح می دهیم.

اگر فرض کنیم P تعداد عاملهای موجود در سیستم مدیریت منابع می باشد و W_{pk} که در آن *(p: ١،٢,...,P)* بار کاری عامل p در گام k باشد. در اینصورت میانگین بار کاری عبارتست از:

$$\overline{W_k} = \frac{\sum_{p=1}^{P} W_{pk}}{P} \qquad (٥-٤)$$

میانگین انحراف معیار W_{pk} که بیان کننده سطح موازنه بار[١] سیستم است، به صورت زیر تعریف می شود:

$$L_k = \sqrt{\frac{\sum_{p=1}^{P} (\overline{W_k} - W_{pk})^2}{P}} \qquad (٦-٤)$$

[١] Load Balancing Level

و در نهایت کارایی[1] روش موازنه بار به صورت زیر تعریف می شود:

$$e_k = \frac{L_0 - L_k}{C_k} \quad (٧-٤)$$

که در آن e_k به معنی کارایی موازنه بار در گام k ام و C_k تعداد کل ارتباطات عاملها برای رسیدن به L_k می باشد.

برای مقایسه کارایی روش ارائه شده با روش اولیه [۳۸] بایستی نسبت $e_{k_{new}} / e_{k_{Trad}}$ را محاسبه کنیم.

به لحاظ ریاضی L_k نشان دهنده میانگین انحرافات عاملها از میانگین کلی $\overline{W}_k$ می باشد. بنابراین هرچه مقدار آن به صفر نزدیکتر باشد، بار سیستم متوازنتر است.

همچنین به منظور سادتر شدن مساله، فرض می کنیـــم که هر گره پس از انجام یک بار فرایند موازنه بار به مقدار میانگین ($\overline{W}_k$) می رسد و نیاز به موازنه بار بیشتری ندارد. یعنی:

$$\overline{W}_k - W_{pk} = 0 \quad (٨-٤)$$

از طرف دیگر پس از k مرحله، اگر ظرفیت حافظه در نظر گرفته شده برای ذخیره سازی اطلاعات گره های پربار و کم بار برابر *'a'* باشد *(a>۲)*، در اینصورت *"ka"* عنصر بالانس شده و $P-ka$ عنصر بالانس نشده داریم. در اینصورت:

$$L_{k_{new}} = \sqrt{\frac{\sum_{p=1}^{P-ka}(\overline{W}_k - W_{pk})^2}{P}} \quad (٩-٤)$$

در حالیکه می دانیم:

$$L_{k_{Trad}} = \sqrt{\frac{\sum_{p=1}^{P-2k}(\overline{W}_k - W_{pk})^2}{P}} \quad (١٠-٤)$$

چون فرض کرده ایم که[2] $a>$ می توانیم نتیجه بگیریم که:

$$P-2k > P-ka \quad (١١-٤)$$

که در آن $P-2k$ تعداد گره های موازنه بار نشده در روش اولیه بوده و $P-ka$ تعداد گره های بالانس نشده در روش جدید. پس از K مرحله تفاوت گره های بالانس شده در این دو روش به صورت زیر در می آید:

$$P-2a-P+ka = k(a-2) \quad (١٢-٤)$$

و داریم:

$$L_{k_{Trad}} = \sqrt{\frac{\sum_{p=1}^{P-ka}(\overline{W}_k - W_{pk})^2}{P} + \frac{\sum_{p=ka}^{P-2k}(\overline{W}_k - W_{pk})^2}{P}} \quad (١٣-٤)$$

$$L_{k_{new}} = \sqrt{\frac{\sum_{p=1}^{P-ka}(\overline{W}_k - W_{pk})^2}{P}} \quad (١٤-٤)$$

در اینصورت $L_{k_{Trad}} > L_{k_{new}}$ و درنتیجه:

$$\frac{L_{k_{new}}}{L_{k_{Trad}}} < 1 \quad (١٥-٤)$$

[1] Efficiency

از رابطه ٤-٧ می توان نتیجه گرفت که:

$$\frac{e_{k_{new}}}{e_{k_{Trad}}} = \frac{2(L_0 - L_{k_{new}})}{L_0 - L_{k_{Trad}}} \longrightarrow \frac{e_{k_{new}}}{e_{k_{Trad}}} > 2 \qquad (١٦-٤)$$

همانطور که از اثبات برمی آید، کارایی روش جدید بیش از دو برابر روش اولیه است.

٤-٣-٣- راهکار موازنه بار در سطح همسایگی با استفاده از تکنیک مسیریابی مجازی

همانطور که قبلاً نیز اشاره شد، این روش با الهام از مکانیزمQLBVR [٩٢] ارائه شده است. البته مکانیزم QLBVRبرای موازنه بار در سیستمهای توزیع شده محاسباتی به وجود آمد. ویژگی منحصر به فرد این روش این است که در سطح زمانبند اجرا شده و بار گره ها را بر اساس دو فاکتور "طول صف " و "نرخ ورود به صف " در دو سطح مجزا از هم موازنه می کند. همچنین این روش از راهکار "مسیریابی مجازی " برای سنجش سودمندی ارسال بار استفاده می کند.

در روش مسیریابی مجازی، همانطور که در بخش ٣-٣-٤ گفته شد، یک گره مجازی که به آن گره مقصد *(d)* گفته می شود به سیستم اضافه شده و از آن پس مساله سودمندی ارسال، به مساله پیدا کردن کوتاهترین مسیر تبدیل می گردد.

قبل از بیان دقیق روش ارائه شده، لازم است به ویژگیهای ARMS که ساختار موازنه بار سطح همسایگی را برای آن مناسب ساخته و به ویژه امکان تعریف دقیقتری از بار را فراهم می آورد، اشاره کنیم.

در ARMS عاملها کارایی منابع خود را با استفاده از PACE بدست آورده و به صورت دوره ای آنها را فقط با همسایه های خود (برای حفظ کارایی) مبادله می کنند [٣٣]. با توجه به فیلد Options در بسته های مبادله شده بین عاملها، این امکان وجود دارد که اطلاعات بار گره ها را به همراه بسته های مبادله شده بین عاملها مبادله کرد. این کار سبب می شود هزینه تبادل اطلاعات در روش سطح همسایگی نا چیز و حتی قابل چشم پوشی باشد.

همانطور که در بخش ٣ اشاره شد، مشکلی که در سنجش میزان بار وجود دارد ناشی از این واقعیت است که قبل از اجرای پردازشها نمی توان زمان لازم برای اجرای آنها را دانست. همچنین میزان بار یک گره را نمی توان تنها با توجه به یک معیار، یعنی تعداد فرایندهای منتظر اجرا در آن سنجید؛ بلکه فاکتورهای دیگری همچون نرخ ورود تقاضا و نرخ تکمیل کار در یک گره نیز بر روی میزان بار آن موثرند. به ویژه این پارامترها در تخمین میزان بار یک گره در آینده نیز موثرند. در [٨٠] یک فاکتور ترکیبی برای اندازه گیری بار ارائه شده است، اما این روش هم در نهایت تعداد کارها را می شمرد که باز همان مشکل قبلی را در بر دارد.

بنابراین، اگر بتوان راهکاری برای پیشگویی صحیح میزان زمان لازم برای اجرای پردازشها در دست داشت، می توان میزان بار صحیح هر گره را یافت و حتی تغییرات آنرا در زمانهای آینده نیز تخمین زد و به این ترتیب دقت الگوریتمهای موازنه بار را افزایش داد. در سیستم ARMS با استفاده از قابلیت PACE می توان به این پیشگویی با دقت خوبی دست یافت. بنابراین برای اندازه گیری و یا تخمین بار بر اساس رابطه ٤-١٧ عمل کرده و داریم:

$$L_i(t+\Delta t) = L_i(t) - C_i(t+\Delta t) * \overline{T} + j_i(t, t+\Delta t) * \overline{T} \qquad (١٧-٤)$$

که در آن $C_i(t+\Delta t)$ نشاندهنده تعداد کارهای تکمیل شده در بازه زمانی $[t,t+\Delta t)$ است. $j_i(t,t+\Delta t)$ نیز تعداد کارهای رسیده شده در همان بازه زمانی است. همچنین $\overline{T}$ بیان کننده زمان اجرای متوسط کارها است (که در اینجا مقداری ثابت در نظر گرفته می شود). $L_i(t)$ نیز مجموع زمان اجرا در لحظه t است که با استفاده از قابلیتهای PACE قابل حصول است.

بنا به این نحوه تعریف بار اطلاعات لازم برای محاسبه بار و تخمین آن که باید بین عاملها مبادله شوند، شامل زمان اجرای کلی[۱] در هر گره (که با استفاده از موتور ارزیابی PACE بدست می آید)، میانگین نرخ ورود کار و نرخ تکمیل کار (با در نظر گرفتن تعداد کارهای وارد شده یا تعداد کارهای تکمیل شده در یک بازه زمانی ثابت در هر گره) می باشند.

این نحوهٔ تعریف بار علاوه بر دقت بیشتر نسبت به روشهای قبل و در نظر گرفتن فاکتورهای مناسب برای اندازه گیری بار، امکان تخمین میزان بار یک گره در آینده را نیز به وجود می آورد. علاوه براین عامل گیرنده می تواند تاخیر زمانی بین ارسال و دریافت اطلاعات را محاسبه کرده و آنرا به عنوان هزینه (زمان) تقریبی ارسال بکارگیرد.

همانطور که گفته شد عاملها در این سیستم به صورت دوره ای و در بازه های زمانی T_s ثانیه که به آن *"بازه زمانی تبادل وضعیت*[۲] *"* گفته می شود، اطلاعات خود را مبادله می کنند. لحظه ای که این اتفاق می افتد را *"مقطع تبادل وضعیت*[۳]*"* گفته می شود. برای کم شدن سربار ارتباطی تبادل اطلاعات، باید تا حد امکان T_s را بزرگ انتخاب کنیم. اما مقادیر T_s خیلی بزرگ، باعث کم شدن دقت اطلاعات می شود. بنابراین باید تعادلی[٤] بین این دو فاکتور انتخاب شود. برای کم شدن تعداد تبادلات و در عین حال حفظ دقت لازم در [۵۵] پیشنهاد شده که اطلاعاتی را مبادله کنیم، که بتوانیم بر اساس آن اطلاعات مبادله شده در زمان t، میزان بار را در زمان $t+\Delta t$ تخمین بزنیم. بر این اساس بازه زمانی T_s را می توان به چند زیر بازه تقسیم کنیم. به این زیر بازه ها *"بازه های تخمین*[۵]*"* گفته می شود.

در شکل ۳-۱۰، T_{n-1} و T_n نشاندهنده مقطع تبادل وضعیت و t_1 و t_2 نشاندهنده مقطع تخمین وضعیت است.

باید توجه داشت که این تبادل وضعیتها جزء سربار روش موازنه بار محسوب نمی شوند، بلکه آنها بطور ذاتی باید برای تبلیغ (معرفی) سرویس ARMS باید انجام شوند. ما بطور زیرکانه ای از این امکان استفاده کردیم و فیلدهای مورد نیاز موازنه بار را نیز به آن اضافه کردیم. بنابراین می توان ادعا کرد که به لحاظ حجم تبادل اطلاعات _ که قسمت اعظمی از سربار هر روش موازنه باری را تشکیل می دهد_ روش ارائه شده کاملاً صرفه جویانه عمل می کند و سربار آن برای سیستم ناچیز است.

با توجه به آنچه گفته شد، تصمیم گیری در مورد موازنه بار به صورت زیر انجام می شود.

با استفاده از رابطه ٤-۱۷ و با بکارگیری اطلاعات بار دریافت شده در T_s، عاملها میزان بار فعلی همسایگانشان را تخمین می زنند. سپس عامل میانگین بار روی همسایه هایش را محاسبه می کند. در اینصورت یک عامل در صورتی خود را "پربار " می نامد که بارش بیش از میانگین بار همسایه هایش باشد. اگر عامل خود را پربار تشخیص داد، باید در مورد موازنه بار خود تصمیم گیری کند. برای این منظور عامل پربار، همسایگانی را که میزان بار آنها به اندازه θ از میانگین کمتر است، *"مجموعه فعال*[٦]*"* می نامد.

---

[۱] Total execution Time
[۲] Status Exchange Interval
[۳] Status Exchange Epoch
[٤] Trade-off
[۵] Estimation Epoch
[٦] Active Set

اینکه یک یا چند همسایه بارشان از میانگین کمتر باشد، شرط لازم برای انتخاب گره مقصد است. برای اینکه یک گره حتماً به عنوان مقصد انتخاب شود، باید شرط سودمندی ارسال نیز ارضاء گردد. این شرط بیان می کند که هزینه ارسال به گره مقصد بعلاوه زمان لازم برای اجرا بر روی گره مقصد باید از زمان لازم برای اجرای بار بر روی گره فعلی زمان کمتری ببرد. همانطور که قبلاً نیز گفته شد، اینگونه سودمندی ارسال را با *"مسیریابی مجازی*[1]*"* بررسی می کنیم. بنابراین فرستنده سعی می کند بارها را به عناصری از مجموعه فعال بفرستد که سودمندی بیشتری را به همراه دارند. این قسمت چون فقط به دنبال مسیر بهینه در بین همسایه های گره (مجموعه فعال) می گردد زمان زیادی نمی برد. زمانبر نبودن قسمتهای مختلف الگوریتم سبب می شود که امکان انجام آن در سطح زمانبند امکان پذیر[2] باشد. در نهایت، الگوریتم موازنه بار کارها را یکی یکی انتخاب می کند، برای ارسال هر یک ابتدا پرسودترین مسیر بین عناصر موجود در مجموعه فعال انتخاب می شود و سپس بار به مقصد انتخاب شده منتقل می شود.

مشکلی که ممکن است اتفاق بیافتد این است که یک گره کم بار در مجموعه فعال دو گره متفاوت قرار بگیرد. در چنین شرایطی ممکن است هر دو گره به طور همزمان شروع به ارسال اضافه بار خود به گره کم بار کنند. این مساله ممکن است تا آنجا ادامه یابد که گره کم بار خود تبدیل به گرهی پربار شود. در اینصورت الگوریتم پایداری خود را از دست می دهد. چون هم زمان زیادی صرف کرده ایم، هم شبکه را اشغال کرده ایم و هم تصمیم گیری اشتباهی کرده ایم. در بدترین حالت این پدیده ممکن است باعث تعویض مکانهای متوالی برای یک یا چند پردازش خاص شود. برای اجتناب از چنین پدیده ای مجدداً از روش قفل گذاری استفاده می کنیم. بنابراین یک گره در هر زمان بارش را فقط به یک همسایه می دهد. بدینوسیله از بروز مشکلات فوق جلوگیری می شود.

با توجه به روش ارائه شده، واضح است که در ابتدا اکثر موازنه بارها بین همسایه ها اتفاق می افتد. اما با گذشت زمان شعاع پراکنده گی بار به ناحیه وسیعتری افزایش می یابد.

٤-٣-٤- نتیجه گیری۱

همانطور که در قسمتهای قبلی اشاره شد، الگوریتم موازنه بار مبتنی بر هوش گروهی مورچه ها هر قدر هم که قوی باشد، به علت یکسری ایرادات ذاتی، نمی تواند بعضی از نیازهای یک روش موازنه بار ایده آل را برآورده کند. روش ارائه شده در فوق نیز گرچه به طور ریاضی اثبات شد که از مدل اولیه بهتر است، ولی باز هم دچار همین ایرادات ذاتی است. در زیر به بررسی دقیقتر این ایرادات پرداخته و در مرحله بعد راه حل مناسبی برای فائق آمدن بر آنها ارائه می دهیم.

الف) عدم عادلانگی، یکی از مهمترین ضعفهای روشهای هوش گروهی است. در این روشها همواره این احتمال وجود دارد که بعضی از گره ها برای مدت زمان زیادی پربار باشند و هیچ مورچه ای به آنها سرکشی نکند، در حالیکه بعضی دیگر دائماً مورد بررسی قرار گرفته و موازنه بار شوند. همانطور که گفته شد، این ایراد به دلیل حرکت تصادفی مورچه ها در محیط رخ می دهد.

ب) کشف وترمیم به جای پیشگیری. قبلاً نیز اشاره شد که روش مورچه ها یک روش کشف وترمیم اختلاف بار سیستم است. اصولا در یک سیستم ایده ال بهتر است اختلاف باری پیش نیاید، که احتیاج به ترمیم آن با تحمیل سربار اضافه به سیستم باشد. به عبارت دیگر احتیاج به یک روشی داریم که تا حد امکان از اختلاف بار پیشگیری کند.

---

[1] Virtual Routing
[2] Feasible

ج) حجم ارتباطات بالا در روش مورچه ها. در مدل استفاده از مورچه ها اگر میزان اختلاف بار زیاد باشد، تعداد مورچه ها و در نتیجه حجم ارتباطات[۱] به شدت افزایش می یابد. بر طبق رابطه ٤-۷ این حجم بالای ارتباطات سبب کاهش کارایی روش می شود. بنابراین اگر بتوان فی البداه اختلاف بار موجود را کاهش داد و سپس از روش مورچه ها به عنوان یک مکمل استفاده کرد، به نظر می آید تعداد مورچه ها کاهش و در نتیجه کارایی افزایش یابد.

با توجه به نقائص فوق در روش مورچه ها، روش "موازنه بار درسطح همسایگی" را به عنوان مکملی برای روش مورچه ها ارائه می دهیم. تعامل این دو روش با هم، باعث به وجود آمدن روشی کارا برای موازنه بار در محیط گرید می شود.

٤-۳-۵- نتیجه گیری۲

مکانیزم موازنه بار سطح همسایگی، با توجه به خواص و ویژگیهایی همچون پیشگویی زمان اجرا و داشتن اطلاعات نسبت به همسایه ها، که در ARMS وجود دارد، باعث می شود تا اضافهٔ بارکاری گره ها در همان ابتدا به گره های مجاور ارسال شود. این نحوه عملکرد سبب پخش ناحیه ای[۲] بار در سطح گرید می شود. به عبارت دیگر چون این روش بار را در سطح همسایه های گره های پربار پخش می کند، مدت زمان زیادی طول می کشد تا این بار در سطح کل گرید پخش شود. پخش ناحیه ای بار توسط روش سطح همسایگی در آزمایشهای بخش بعدی نیز به وضوح دیده می شود. بنابراین می توان گفت روش سطح همسایگی یک روش سراسری محسوب نمی شود. این مساله تفاوت (ضعف) اصلی روش فعلی در برابر روش مبتنی بر گروه مورچه ها است.

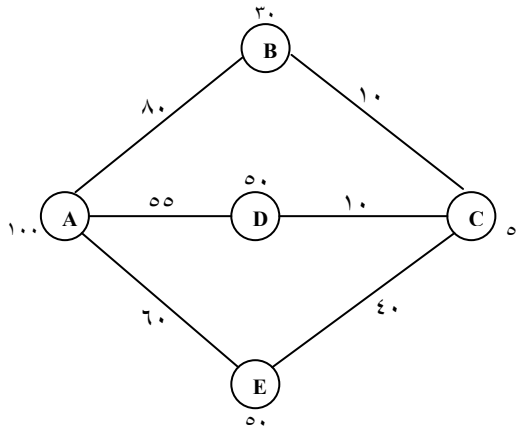
ضعف دیگری که در روش موازنه بار سطح همسایگی وجود دارد دامنه دید محدود آن است. این مساله بعضاً باعث دور شدن از تصمیم بهینه توسط گره فرستنده می شود. به عنوان مثال در شکل ٤-۵، A گرهی پربار است و آخرین برنامه باید ۱۰۰ ثانیه برای اجرا انتظار بکشد. طبق الگوریتم ارائه شده A نمی تواند بارش را به هیچ یک از همسایه هایش بفرستد. اما اگر مسیر $A \rightarrow B \rightarrow C$ کشف شود، بار به سرعت در محیط پخش می شود.

همانطور که دیده می شود، راه حل هر دو نقیصه فوق، تعمیم الگوریتم در یک دامنه وسیع (در حالت حدی در سطح کل گرید) است. اما این کار موجب عدم قیاس پذیری روش ارائه شده می شود. عدم قیاس پذیری به دو علت رخ می دهد. اول اینکه جمع آوری حجم زیادی از اطلاعات در هر گره، به علت وسعت زیاد و فرکانس بالای تغییر اطلاعات در گرید مقدور نیست. دوم اینکه الگوریتم مسیریابی مجازی برای تعداد گره های زیاد، زمان زیادی می گیرد. با توجه به این مساله که الگوریتم موازنه بار به ویژه آنهایی که در سطح زمانبند اجرا می شوند، بایستی تا حد امکان سریع و کارا باشند و با توجه به زمان بر بودن عوامل ذکر شده، تعمیم الگوریتم سطح همسایگی در سطح وسیع منطقی به نظر نمی رسد.

مشکل سومی که در این روش وجود دارد این است که ممکن است پس از عملیات موازنه بار برای یک گره، هنوز آن گره پربار باشد. این مشکل نیز ناشی از محدود بودن دامنه پخش بار در سطح عناصر مجموعه فعال است.

---

[۱] Communication
[۲] Regional



شکل ٤-٥. نحوه عملکرد الگوریتم سطح همسایگی.

٤-۳-٦- راهکار موازنه بار چند سطحی(MLBM[1])

با توجه به جزئیات ارائه شده برای هر یک از روشهای بخشهای ٤-۳-۱ و ٤-۳-۲ و همچنین نکات قوت و ضعف هر یک از آنها، می توان به این نتیجه رسید که الگوریتم سطح همسایگی، بار را به صورت ناحیه ای پخش می کند در حالی که روش مبتنی بر گروه مورچه های هوشمند، سعی در پخش بار در سطح کل گرید دارد. به عبارت دیگر شعاع پخش، در روش سطح همسایگی ناحیه ای و در روش گروه مورچه ها سراسری است. منظور از ناحیه ای بودن پخش این است که روش سطح همسایگی تمرکز بار در یک نقطه (گره) را تبدیل به تمرکز بار در یک ناحیه (با چگالی کمتر) می کند. البته دامنه این ناحیه مشخص نیست و در زمان بی نهایت، این دامنه به سمت کل گرید میل می کند. می توان این نحوه موازنه بار را به یک حرکت موجی شکل با سرعت انتشار کم تشبیه کرد. همچنین همانطور که در بخش قبل نیز توضیح داده شد، به علت دامنه محدود گره های مطلع از هم (تا سطح همسایه ها)، روش ارائه شده نمی تواند خارج از مرزهای خودش (همسایه ها) راه حلی پیدا کند. در حالی که ممکن است چنین راه حلی وجود داشته باشد.

در مقابل، روش مورچه ها نیز ضعفهای خاص خودش را دارد. مهمترین این ضعفها تعداد زیاد مورچه ها، به خصوص در شرایط نامتوازنی شدید بار در کل سیستم است. در این زمان که احتمالاً کل سیستم نیز به علت وجود بارکاری زیاد در سطح آن، شرایط ترافیکی سنگینی دارد، وجود تعداد مورچه های زیاد نیز بر شدت این ترافیک (چه به لحاظ بار پردازشی آنها و چه به لحاظ میزان ارتباطات) افزوده و باعث سنگینتر شدن بار کل سیستم و افت کارایی کل سیستم می شود. که این پدیده مغایر با هدف الگوریتم موازنه بار است.

با توجه به این نکات، ترکیب دو روش ارائه شده در قالب یک روش چند سطحی می تواند راه حل خوبی باشد. در اینصورت این دو روش با کمک هم نقائص هم را رفع کرده و روش ایده الی را به وجود می آورند که اکثر اهداف بیان شده در بخش ۳-۲-۱ را برآورده می کنند. آزمایشات انجام شده در بخشهای بعد این مساله را تایید می کند. بر این اساس، نحوه همکاری قسمتهای مختلف در سیستم MLBM را می توان به صورت زیر بیان کرد.

هر گره به صورت دوره ای[2] عمل جمع آوری اطلاعات همسایه ها را انجام داده و براساس الگوریتم سطح همسایگی سعی در موازنه بار خود می کند. همزمان، ممکن است مورچه هایی نیز از آن گره عبور کنند و در صورتیکه آن گره را پربار یا کم بار تشخیص دادند، مشخصات آنرا به حافظه خود بسپارند، تا در مراحل بعدی بار آن را موازنه کنند. البته

[1] Multi-level Load Balancing Mechanism
[2] Periodic

همانطور که قبلاً گفته شد این امکان وجود دارد که به صورت اتفاقی هیچ مورچه ای از آن گره عبور نکند. در چنین شرایطی اگر مورچه برای چند دوره متوالی حتی با اعمال الگوریتم سطح همسایگی هنوز پربار باقی بماند (در میان همسایه های خود برای چند دوره متوالی بیش از میانگین بار داشته باشد و یا پس از اعمال روش سطح همسایگی هنوز پربار باشد)، طبق آنچه در الگوریتم مورچه ها گفته شد، خود گره می تواند مورچه ای ایجاد کرده و آن را به سیستم بفرستد. برای اینکه این مورچه سریع موازنه بار را انجام دهد، معمولاً تعداد گامهای آن کم انتخاب می شود و در صورتیکه کل سیستم در حالت متوازنی باشد در دوره های بعدی طول عمر آن افزایش می یابد. لازم به ذکر است که همزمان با این دو الگوریتم، الگوریتم دیگری (الگوریتم مبتنی بر ژنتیک الگوریتم که از این پس از آن تحت عنوان *"سطح محلی"* یاد می کنیم) [٤٠] در سطح هر گره، سعی در ایجاد یک زمانبندی بهینه برای هر گره می کند تا از منابع (پردازنده های) آن به نحو مطلوبتری استفاده شود.

باید توجه داشت که طبق آنچه بیان شد، شعاع پخش الگوریتم سطح همسایگی با سرعت کمی رشد می کند. یعنی با استفاده از این الگوریتم زمان زیادی طول می کشد که بار در کل سیستم (در سطح ناحیه های بزرگ) پخش شود. این ضعف با بکارگیری الگوریتمی که سطح تحت پوشش آن سراسری است باید جبران شود. با توجه به خواص و نحوه عملکرد گروه مورچه های هوشمند که در بخش ٤-۳-۱ ارائه شد، می توان از آن برای نیل به این هدف استفاده کرد. البته همانطور که اشاره شد، ضعف روش مورچه ها در افزایش تعداد مورچه ها، میزان ارتباطات آنها و در نتیجه زمان لازم برای پردازش مورچه ها است. اینها همه باعث ایجاد سربار برای سیستم و افت کارایی الگوریتم موازنه بار می شوند. در استفاده همزمان الگوریتم مورچه ها و الگوریتم سطح همسایگی، الگوریتم سطح همسایگی بار را تا حد زیادی در محیط پراکنده کرده و سیستم را به حالت بالانس نزدیک می کند. این شرایط منجر به بکارگیری تعداد مورچه های بسیار کمتری برای موازنه بار می شود. در نتیجه میزان ارتباطات کاهش یافته و کارایی الگوریتم موازنه بار و کل سیستم افزایش می یابد. بدین ترتیب استفاده همزمان دو الگوریتم، سبب مرتفع شدن عیب روش مورچه ها نیز می شود.

با جمع بندی نکات بیان شده و بررسی مزایا و معایب هر یک، و همچنین چگونگی مرتفع شدن این معایب در مدل MLBM، می توان گفت که روش سطح همسایگی روشی است که در سطح گره و به طور دائم اجرا می شود و بار را به صورت ناحیه ای پخش می کند و در عین حال بار کمی را به سیستم تحمیل می کند. در سطحی دیگر الگوریتم مورچه ها قرار دارد که به صورت مستقل از گره ها بوده، تصادفی عمل کرده و بار را به صورت سراسری پخش می کند. در سطح هر گره هم الگوریتم سطح محلی قرار دارد که به صورت محلی (در سطح گره) عمل کرده و بار هر گره را موازنه می کند.

همانطور که بیان شد راهکار MLBM اکثر ویژگیهای اشاره شده در بخش ۳-۲-۱ را برآورده می کند. در این قسمت به بحث در این مورد پرداخته و سعی در بیان دلائل برآورده شدن ویژگیهای ذکر شده در آن می کنیم.

اول اینکه مدل ارائه شده قیاس پذیر است. برای اثبات این مساله کافیست ثابت کنیم هر کدام از سطوح قیاس پذیرند یا به عبارت دیگر کاراطی آنها به تعداد گره ها بستگی ندارد. روش سطح همسایگی با تغییر اندازه سیستم و اضافه یا کم شدن گره ها دچار افت کارایی نمی شود. دلیل این امر بیشتر در ذات توزیع شده بودن الگوریتم است. همچنین به علت اینکه روش فقط در سطح همسایه ها انجام می شود، با اضافه شدن گره جدید میزان تبادلات (جز همان مبادلات لازم برای ARMS) و یا پردازشهای لازم برای انجام موازنه بار زیاد نمی شود. اما روش مبتنی بر گروه مورچه های هوشمند اگر به تنهایی برای موازنه بار استفاده شود، در شرایطی که اختلاف بار در سیستم زیاد باشد، منجر به انفجار جمعیت مورچه ها و افت کارایی و در نتیجه عدم قیاس پذیری می شود؛ اما در ترکیب دو روش، چون بارکاری در همان ابتدای

کار تا حد زیادی توسط الگوریتم سطح همسایگی پخش می شود، تعداد مورچه های کمی به وجود می آیند و در نتیجه قیاس پذیری مدل ارائه شده حفظ می شود.

MLBM الگوریتمی تطبیق پذیر است. منظور از تطبیق پذیر بودن این است که مدل به سرعت خود را با شرایط محیطی وفق دهد. در یک سیستم موازنه بار این شرط را می توان اینگونه تفسیر کرد که سیستم مورد نظر در شرایط اختلاف بار، به سرعت آنرا برطرف کند. در عین حال در شرایط نزدیک به حالت تعادل هم الگوریتم باید در لاک خود فرو رفته و بار کمی را به سیستم تحمیل کند. کلیه این اعمال در مدل ارائه شده انجام می شوند. برای اینکه سیستم به سرعت به تغییرات واکنش دهد بایستی به طور دائم به وضعیت گره ها سرکشی کند، اگر اختلاف باری وجود داشت تا حد امکان خودش آنرا برطرف کند. این قسمت دقیقاً توسط الگوریتم سطح همسایگی انجام می شود. بنابراین سیستم سریعاً به تغییرات بار واکنش نشان می دهد. همانطور که بیان شد، الگوریتم مبتنی بر هوش گروهی نیز عملیات خود شامل تصمیم گیری در مورد وضعیت گره، تغییر جمعیت مورچه ها و تغییر تعداد گامهای مورچه را کاملاً منطبق بر شرایط سیستم انجام می دهد.

MLBM الگوریتمی پایدار است. یعنی تا حد امکان تصمیمات نادرست نمی گیرد. همانطور که در قسمت ٤-۳-۲ بیان شد، اگر الگوریتم سطح همسایگی به تنهایی استفاده شود، این امکان وجود دارد که بعضاً از تصمیمات بهینه دور شود. اما با کمک الگوریتم مورچه ها این مشکل برطرف می گردد. نکته دیگری که باعث عدم پایداری می گردد از بین رفتن صحت اطلاعات گره ها به ویژه در روش گروه مورچه ها می باشد. همانطور که در ٤-۳-٤ شرح داده شد این مساله نیز با تکنیک قفل گذاری تا حد زیادی برطرف شده است. روش جدید اندازه گیری بار نیز باعث دید دقیقتر گره ها از میزان بار هم گشته و در برقراری پایداری سیستم کمک می کند.

MLBM شفاف از دید کاربر عمل می کند. در الگوریتم موازنه بار مبتنی بر گروه مورچه های اولیه [۳۸] این شفافیت وجود نداشت و خود کاربر می بایست در مورد میزان جمعیت مورچه ها تصمیم گیری می کرد. اما در هر دو سطح سیستم ارائه شده، مساله شفافیت بطور کامل برقرار است و همه کارها - همانطور که در محیط گرید انتظار می رود- بدون دخالت کاربر و پنهان از کاربر انجام می شود.

تحمل پذیری در برابر خطا نیز توسط الگوریتم مورچه ها تضمین می شود. در شرایطی که به هر علت برای گرهی خطایی پیش بیاید و بخواهد از سیستم خارج شود، عامل مربوطه به سرعت مورچه ای با تعداد گامهای کم ایجاد کرده و آن را در سیستم رها می کند تا بار گره خطا دار را در سیستم پخش کند.

و در نهایت MLBM سربار بسیار کمی را به سیستم تحمیل می کند. دلیل این امر این است که اولاً الگوریتم سطح همسایگی بار ناچیزی را به سیستم تحمیل می کند (به علت سوار شدن بر روی ساختارهای مبادلاتی در ARMS)؛ ثانیاً در الگوریتم مبتنی بر مورچه ها نیز همانطور که گفته شد در زمانیکه مورچه ها احساس کنند فایده ای برای موازنه بار سیستم ندارند، دست به خودکشی می زنند تا بار کمتری به سیستم تحمیل شود.

٤-٤- محیط شبیه سازی

در این تحقیق، سیستم مدیریت منابع ARMS و روش ارائه شده به صورت ساده شده و با در نظر گرفتن جنبه های لازم که پارامترهای موثر سیستم را نشان می دهند شبیه سازی شده است. قسمتهای مختلف این محیط به صورت زیر شبیه سازی شده اند:

- *عاملها*. عاملهای سیستم را می توان به صورت یک گراف نشان داد. این ساده سازی در چند کار مشابه دیگر نیز انجام شده است [۸۲، ٤۰، ۳۸]. هر عامل دارای تعدادی عامل همسایه است. تمام آزمایشاتی این پایان نامه با ٤۰۰ گره (عامل) انجام شده است.
- *بار کاری*[۱]. یک مقدار به عنوان بارکاری به هر گره تخصیص داده می شود. این مقدار در ابتدا به صورت تصادفی[۲] به هر گره داده می شود. این مقدار در طول زمان با توجه به نرخ ورود و نرخ تکمیل کار در هر گره، تغییر می یابد.
- *نرخ ورود*[۳] *و نرخ تکمیل*[٤] *کار*. در هر دوره، نرخ ورود و نرخ تکمیل کار گره به صورت تصادفی به هر عامل اختصاص می یابد.

همچنین به منظور انجام آزمایشات و مقایسه با سایر روشهای قبلی، علاوه بر پیاده سازی ایده های جدید، الگوریتم مبتنی بر گروه مورچه های اولیه [۳۸] و نیز الگوریتم *"بهترین برازش"*[۵] [۸۳] (جهت مقایسه با روش سطح همسایگی) نیز بر روی سیستم ARMS پیاده سازی شده اند.

پیاده سازی محیط شبیه سازی و الگوریتم های موازنه بار توسط زبان برنامه نویسی VB.Net بوده است. در طراحی این محیط سعی شده است تا اصول برنامه نویسی پیمانه ای کاملاً رعایت شود تا در صورت ایجاد تغییرات در ساختار الگوریتم به راحتی تغییرات مورد نظر اعمال گردند. تمامی عملکردهای مورد نیاز بصورت کلاسهای جداگانه پیاده سازی شده اند و تا حد امکان اصول شیئ گرایی رعایت شده اند تا بتوان به راحتی آنها را با نسخه های بعدی (در صورت نیاز) تعویض نمود. ارتباطات مابین تمامی قسمتها از طریق ارسال پیامهای تعریف شده، انجام می‌شود که وابستگی ساختاری قسمتهای مختلف به یکدیگر را به حداقل می رساند.

٤-۵- آزمایشهای انجام شده

همانطور که در قسمتهای قبلی این بخش اشاره شد، هر یک از روشهای ارائه شده (روش مبتنی بر گروه مورچه های هوشمند و روش سطح همسایگی)، به تنهایی و مجزا از دیگری یک روش موازنه بار محسوب می شوند و خواص و ویژگیهای خاصی را از خود بروز می دهند. همچنین ادعا شد که ترکیب این دو روش در ساختار چند سطحی MLBM، باعث بهبود کارایی هر یک از آنها و در نهایت کارکرد بهینه کل روش می شود.

در این قسمت سعی داریم با استفاده از شبیه سازی و نتایج آزمایشهای مختلف انجام گرفته به اثبات این ادعاها بپردازیم. هر چند در بخش ٤-۳-۲، بهبود کارایی روش مبتنی بر گروه مورچه های هوشمند در مقایسه با روش اولیه [۳۸] به صورت ریاضی اثبات شد، با این حال در این قسمت نشان می دهیم که نتایج تجربی بدست آمده نیز آن اثبات را به صورت عملی تایید می نمایند. برای این منظور، آزمایشات انجام گرفته روی هر یک از روشها و همچنین ترکیب آنها را در سه بخش مجزا طبقه بندی کرده ایم.

---

[۱] Workload
[۲] Random
[۳] Arrival Rate
[٤] Completion Rate
[۵] Best Fit

در قسمت اول، پس از پیاده سازی روش مبتنی بر گروه مورچه ها هوشمند و همچنین روش مبتنی بر گروه مورچه های اولیه در محیط ARMS، آزمایشهای متنوعی بر روی نحوه کارکرد و کارایی آنها انجام داده و آنها را با هم مقایسه کرده ایم.

در قسمت دوم به بررسی تاثیر کارکرد روش سطح همسایگی در موازنه بار نهایی سیستم پرداخته و این تاثیرات را از جهات مختلف با مدلهای موازنه بار پایه مقایسه می کنیم. چون در مدل چند لایه MLBM، وظایف روش سطح همسایگی و روش مبتنی بر گروه مورچه های هوشمند از هم مجزا بوده و در دو سطح مختلف قرار می گیرند، در این قسمت در پی مقایسه کارایی این دو روش با هم نیستیم. بلکه سعی در مقایسه کارایی روش سطح همسایگی، با روشهای مشابه موجود می کنیم. یکی از روشهای مشابه، الگوریتم بهترین برازش [۸۳] است. در این الگوریتم در هر دوره، گره پربار، اضافه بار خود را به کم بارترین (بهترین) همسایه خود می فرستد. برای انجام مقایسه بین این دو روش، هم روش سطح همسایگی و هم روش بهترین برازش در محیط ARMS شبیه سازی گریده اند.

در قسمت آخر نیز به بررسی کارایی مدل چند سطحیMLBM پرداخته و کارایی آنرا به ویژه با حالتی که تنها از روش مبتنی بر گروه مورچه ها استفاده کنیم، مقایسه کرده ایم. نتایج قسمت پایانی نشان می دهند که روش چند سطحی MLBM، نسبت به تک تک روشهای دخیل در آن، یعنی روش مبتنی بر گروه مورچه های هوشمند و روش سطح همسایگی، و همچنین نسبت به روشهای قبلی، یعنی روش مبتنی بر مورچه اولیه کارایی بهتری را ارائه می دهد.

٤-٥-١- بررسی اثر تعداد مورچه های ایجاد شده بر سطح موازنه بار

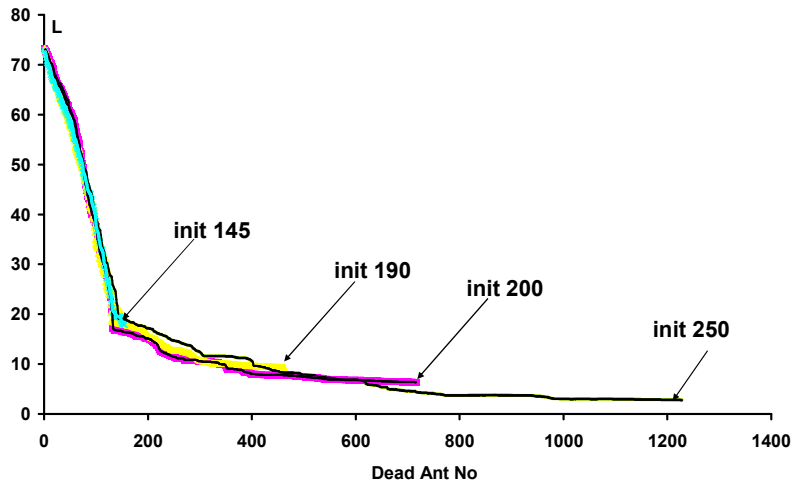
معرفی

در این آزمایش رابطه بین سطح موازنه بار و تعداد مورچه های به وجود آمده را در مکانیزم مبتنی بر دسته مورچه های هوشمند بررسی کرده ایم. همانطور که قبلاً ادعا شد، تعداد مورچه های بیشتر، باعث موازنه بار بیشتری می شوند. برای اثبات این مدعا در این آزمایش نسبت تعداد مورچه های به وجود آمده*(Dead Ant No)* به سطح موازنه بار ایجاد شده*(L)* را سنجیده ایم. "سطح موازنه بار[۱] "را میانگین انحراف کل عاملها از میانگین بار سیستم تعریف کرده و مقدار آن طبق رابطه ٤-٦ محاسبه می شود. بنابراین هرچه مقدار سطح موازنه بار به صفر نزدیکتر باشد، سیستم به حالت بالانس نزدیکتر است. این آزمایش به ازای مقدار حافظه $a=۷$ و تعداد گامهای اولیه $S=۱۵$ برای هر مورچه انجام شده است.

تحلیل و بررسی

همانطور که در شکل ٤-٦ دیده می شود، آزمایش با تعداد مورچه های متفاوت در ابتدای کار*(init)* انجام شده است. با افزایش تعداد مورچه ها سطح موازنه بار سیستم نیز بهبود یافته است. البته باید توجه داشت که تعداد مورچه های بیشتر منجر به حجم ارتباطات بیشتری نیز می گردد.

[۱] Load Balancing Level



شکل٤-٦. تاثیر تعداد مورچه ها بر روی میزان موازنه بار

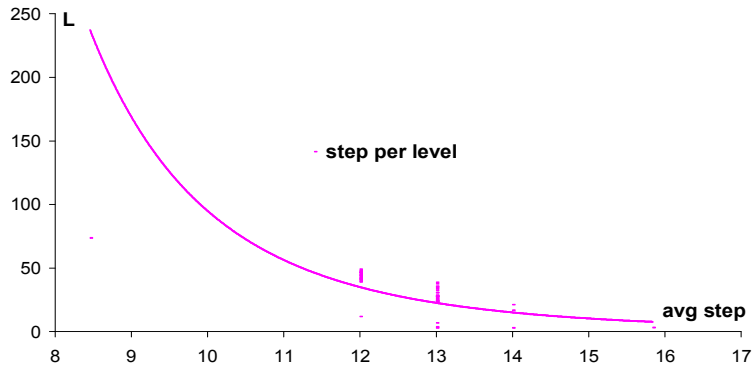
٤-٥-٢- بررسی رابطه بین تعداد گامهای مورچه و سطح موازنه بار

معرفی

همانطور که گفته شد، با نزدیک شدن سیستم به همگرایی نهایی، تعداد گامهای مورچه ها افزایش می یابد. در این آزمایش به بررسی صحت این گفته پرداخته ایم. برای این منظور، میانگین تعداد گامهای مورچه ها *(Avg Step)* را در زمانهای مختلف نسبت به سطح موازنه بار *(L)* مورد سنجش قرار داده ایم.

تحلیل و بررسی

با گذشت زمان و عملکرد مورچه ها در سیستم، سیستم به حالت همگرایی نهایی نزدیکتر می شود، بنابراین طبق آنچه گفته شد، انتظار می رود که طول گامهای مورچه های سیستم به طور متوسط افزایش یابد. این نحوه عملکرد در شکل ٤-٧ نشان داده شده است.

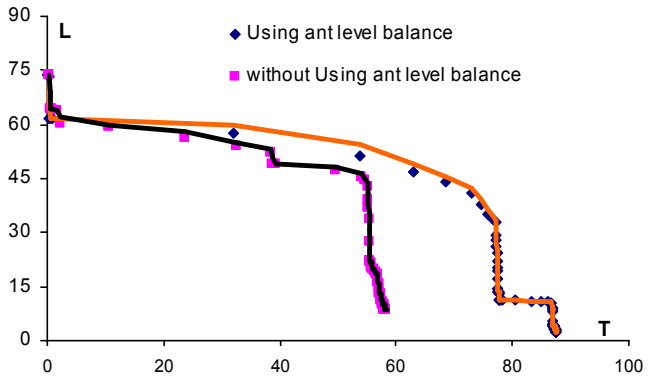


شکل ٤-٧. رابطه بین میانگین تعداد گامهای مورچه و سطح موازنه بار حاصل

٤-٥-٣- بررسی تاثیر عملکرد "موازنه بار در سطح مورچه ها"

معرفی

در این آزمایش تاثیر عملکرد ایده "موازنه بار در سطح مورچه ها " بر روی سطح موازنه بار نهایی نشان داده شده است. برای انجام این آزمایش،که در شکل ٤-٨ نشان داده شده است، سطح موازنه بار در طول زمان عملکرد سیستم یک بار با استفاده از ایدهٔ موازنه بار در سطح مورچه ها و بار دیگر بدون استفاده از آن اندازه گیری شده است.



شکل ٤-٨ مقایسه تاثیر استفاده از موازنه بار در سطح مورچه، بر روی سطح موازنه بار نهایی سیستم *(L)* در طول زمان *(T)* .

تحلیل و بررسی

همانطور که از شکل ٤-٨ مشخص است، استفاده از روش موازنه بار در سطح مورچه ها، منجر به موازنه بهتر بار در سطح سیستم در طول زمان می شود. البته واضح است که این بهینگی در ازای صرف هزینه زمانی بیشتر حاصل شده است. افزایش هزینه زمانی در شکل ٤-٨ مشخص است. همچنین باید اشاره کرد که استفاده از این روش باعث می شود در حالی که مورچه در دامنه محدودی حرکت می کند، اثر گذاری آن سراسری باشد. همچنین استفاده از این روش اعتبار اطلاعات مبادله شده تضمین شده است، در حالی که در روش استفاده از اسید مورچه، حتی با استفاده از خاصیت تبخیر نیز صحت اطلاعات تضمین شده نیست.

٤-٥-٤- بررسی کارایی مدل مبتنی بر دسته مورچه های هوشمند

معرفی

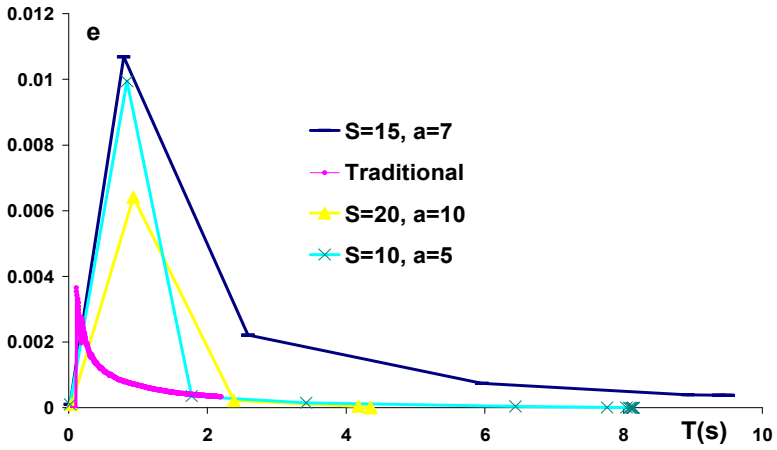
در این آزمایش به مقایسه کارایی روش مورچه های هوشمند ارائه شده و روش مبتنی بر مورچه های اولیه می پردازیم. همچنین در این آزمایش کارایی روش ارائه شده، با تعداد گامهای اولیه متفاوت و همچنین میزان حافظه متفاوت برای مورچه ها نیز مورد آزمایش قرار می گیرد. برای انجام این آزمایش، میزان کارایی *(e)* با توجه به رابطه ٤-٧ در زمانهای مختلف *(T)* سنجیده شده است.

تحلیل و بررسی

همانطور که در شکل ٤-٩ دیده می شود، الگوریتم جدید در هر حالت بهتر از الگوریتم اولیه عمل می کند. البته با توجه به اثبات انجام گرفته در بخش ٤-٣-٢ این امر قابل پیش بینی بود. در واقع این آزمایش اثبات انجام شده را به صورت عملی نیز تایید می کند.

همچنین در این آزمایش، کارایی الگوریتم مورچه های هوشمند با تعداد گامها و میزان حافظه های متفاوت مورد مقایسه قرار گرفته اند. چون مورچه ها به لحاظ تعداد گام، تطبیق پذیر عمل می کنند، (یعنی با توجه به شرایط محیطی تعداد گامهای خود را تنظیم می کنند) اینکه تعداد گامهای اولیه آنها چقدر باشد خیلی مهم نیست. در واقع پارامتر مهم و تاثیرگذار بر روی کارایی، میزان حافظه مورچه است که در اینجا مورد بحث قرار می گیرد. در صورتیکه میزان حافظه

مورچه زیاد باشد (مثلاً $a=۱۰$)، دیرتر اشباع شده و در نتیجه احتمال به وجود آمدن مورچه های جدید کمتر می شود. با کم شدن تعداد مورچه ها، احتمال اینکه هر گره بیش از یک بار بالانس شود کم می شود و این مساله باعث تاخیر در زمان رسیدن به حالت همگرایی می شود. در نتیجه میزان L_k-L کم می شود و در نهایت میزان کارایی کاهش می یابد. در طرف دیگر اگر میزان حافظه مورچه کم باشد (مثلا $a=۵$)، تعداد زیادی مورچه تولید خواهد شد، این مساله با آنکه منجر به همگرایی سریعتر الگوریتم می گردد ولی از طرف دیگر میزان ارتباطات (C) را به شدت افزایش می دهد. با زیاد شدن میزان ارتباطات، طبق رابطه ٤-٧ و همچنین شکل٤-٩، میزان کارایی کاهش می یابد. همانطور که از شکل٤-٩ نیز برمی آید، میزان حافظه باید به گونه ای باشد که در عین کم بودن میزان ارتباطات (C)، تعداد مناسبی مورچه نیز تولید شوند تا با کارایی بالایی بار سیستم را متوازن کنند.

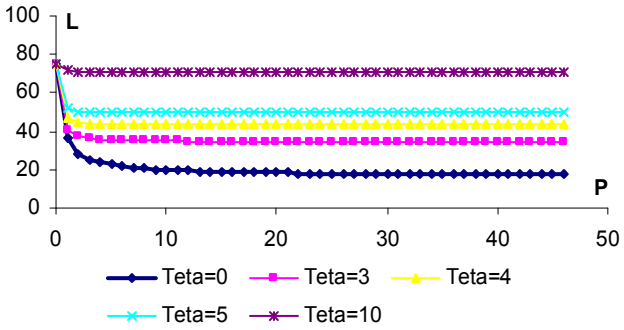


شکل ٤-٩. مقایسه میزان کارایی روش اولیه با روش مبتنی بر مورچه های هوشمند با در نظر گرفتن تعداد گامها (S) و میزان حافظه متفاوت (a) در ابتدای کار.

در ادامه آزمایشهای مربوط به جزئیات نحوه و کارایی عملکرد روش موازنه بار سطح همسایگی ارائه می شود.

٤-٥-٥- بررسی رابطه بین تعداد دوره ها و سطح موازنه بار در الگوریتم سطح همسایگی معرفی

هدف از انجام این آزمایش، که در شکل ٤-١٠ نشان داده شده است، بیان نحوهٔ عملکرد الگوریتم سطح همسایگی است. در این آزمایش سطح موازنه بار حاصل (L) در گذر زمان (در طی دوره ها) (P) در شرایط متفاوت مورد بررسی قرار گرفته است.



شکل ٤-١٠. رابطه سطح موازنه بار (L) حاصل در گذر زمان

تحلیل و بررسی

همانطور که در شکل٤-١٠ دیده می شود، با گذشت زمان سطح موازنه بار سیستم (L) در نقطه خاصی به ثبات می رسد. میزان پخش شدن بار در سیستم بستگی به اندازه θ دارد، مسلماً مقادیر θ ی کمتر، باعث آزادی بیشتر در پخش بار گره ها گشته و باعث کاهش سطح موازنه بار (همگرایی بیشتر) می گردد. حدی که سطح موازنه بار از آن کمتر نمی شود، همان نقطه ای است که یک گره ترجیه می دهد بجای ارسال بار خود به سایر گره ها، آنها را به صورت محلی (بر روی خود) پردازش کند. به عبارت دیگر با گذشت زمان (P)، یک توزیع بار موجی شکل در سیستم پدید می آید. این مساله در جدول٤-١ نیز نشان داده شده است. همانطور که در این جدول مشخص است، در دوره های اول به علت اینکه بار گره ها نامتوازن است و بعضی گره ها خیلی پربار و بعضی دیگر خیلی کم بارند، در اثر اجرای الگوریتم موازنه بار، سطح موازنه بار به شدت کاهش می یابد (سیستم به سمت توازن نهایی می رود). در حالی که در گامهای آخر، به علت اینکه اکثر گره ها به میانگین همسایه هایشان نزدیک شده اند و بار تا حد زیادی در سیستم پخش شده است، تاخیر زمانی[1] انتقال بار بیش از زمان اجرای محلی آن گشته و دامنه تغییرات سطح موازنه بار (L) کم می شود.

سطح موازنه بار (L)	دوره (P)
٧٨،٦٠٩٢٥٥١٨	٠
٣٩،٩٠٥٠٧٤٨٧	١
٣٣،٤٤٢٤٨٧٩٥	٢
٢٥،١١٨٢٧٠٢٤	٥
٢٢،٧٣٥٨١٩٧٦	١٠
٢١،١٢٨٤١٦٨٨	١٥
١٩،٣٨٨٤٦٣٠٦	٣٠
١٨،٧٩٨٩٣٦١٤	٤٥
١٨،٧١٩١٠٧٨٨	٥٠

جدول٤-١. رابطه بین میزان پخشی بار به دوره های زمانی.

٤-٥-٦- بررسی تاثیر مسیریابی مجازی در الگوریتم سطح همسایگی

معرفی

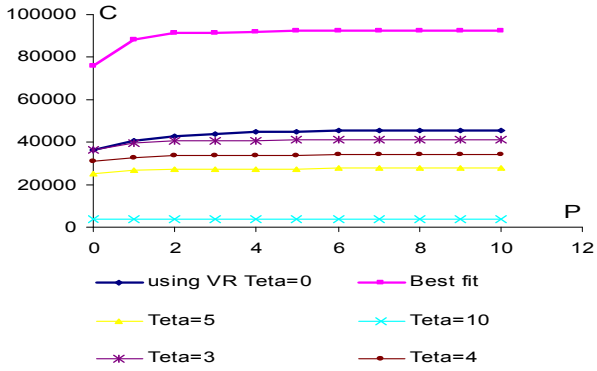
در این آزمایش تاثیر الگوی مسیریابی مجازی را در صرفه جویی زمان صرف شده برای موازنه بار، بررسی کرده ایم. برای به وجود آمدن امکان مقایسه، الگوریتم *"بهترین برازش"* را به عنوان نمونه ای از الگوریتمهایی که زمان انتقال را در نظر نمی گیرند، پیاده سازی کرده ایم. در این آزمایش کل زمان صرف شده برای ارسال اضافه بار به همسایه ها در هر دوره (C) را به عنوان مقیاس در نظر گرفته ایم.

تحلیل و بررسی

همانطور که در شکل٤-١١ دیده می شود این آزمایش به ازای مقادیر مختلف θ انجام شده که در تمامی حالات، هزینه زمانی کمتری نسبت الگوریتم بهترین برازش صرف شده است. مسلماً این امر به دلیل در نظر گرفتن کمترین هزینه

[1] Transmission Cost

ارسال علاوه بر در نظر گرفتن کم بارترین همسایه می باشد در حالیکه در الگوریتم بهترین برازش تنها همسایه ای با کمترین بار در نظر گرفته شده است. همچنین تکرار آزمایش به ازاء مقادیر متفاوت θ، نشان می دهد که مقادیر θ بزرگتر باعث صرفه جویی زمانی بیشتری می شوند. مسلماً این امر به این دلیل رخ می دهد که مقادیر بزرگ θ ارسال بار به بسیاری از گره ها را محدود (منع) می کنند. بنابراین به لحاظ هزینه زمانی به صرفه ترند.



شکل٤-١١. مقایسه سربار (هزینه) زمانی الگوریتم سطح همسایگی با الگوریتم بهترین برازش.

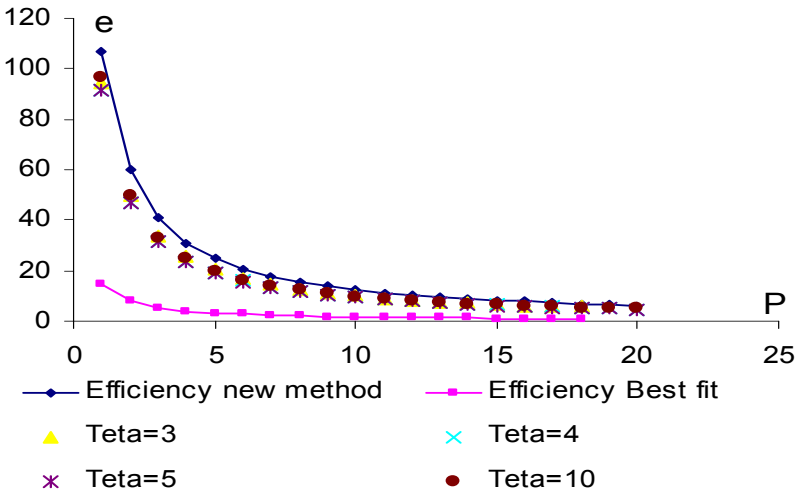
٤-٥-٧- بررسی کارایی الگوریتم سطح همسایگی

معرفی

در این آزمایش کارایی الگوریتم سطح همسایگی به ازاء مقادیر مختلف θ با الگوریتم بهترین برازش مقایسه شده است. برای انجام این آزمایش میزان کارایی (e) که بر اساس رابطه ٤-٧ بدست آمده است، در دوره های مختلف (P) محاسبه شده است.

تحلیل وبررسی

از رابطه ٤-٧ مشخص است که در دوره k ام، مقادیر کمتر C (حجم ارتباطات کمتر) منجر به کارایی بهتری می شوند. اما باید توجه کرد که سطح موازنه بار در دوره k ام (L_k) هم اهمیت دارد. البته این که روش سطح همسایگی منجر به کارایی بهتری می شود، با کمی دقت در آزمایش قابل پیش بینی بود. با این حال شکل٤-١٢ مشخص می کند که در حالی که هزینه ارتباطات (C_k) از الگوریتم بهترین برازش خیلی کمتر است، تفاوت زیادی در سطح موازنه بار حاصله (L_k) در دو روش وجود ندارد.



شکل٤-١٢. مقایسه کارایی الگوریتم سطح همسایگی و بهترین برازش.

همچنین در شکل٤-۱۲ دیده می شود که کارایی میزان θ تاثیر زیادی بر روی کارایی ندارد. به عبارت دیگر هر چه میزان θ بیشتر باشد، میزان تبادلات (C) کاهش می یابد ولی همزمان با آن سطح موازنه بار (L) نیز کم می شود و بالعکس. این امر سبب می شود که میزان کارایی به ازاء مقادیر مختلف θ تقریباً ثابت باشد.

در ادامه آزمایشهای انجام شده بر روی روش MLBM را ارائه کرده و در طی آنها نشان خواهیم داد که این روش از تک تک روشهای ارائه شده تا کنون، یعنی روش مبتنی بر گروه مورچه های هوشمند، روش سطح همسایگی و روش مبتنی بر گروه مورچه های اولیه بهتر عمل می کند.

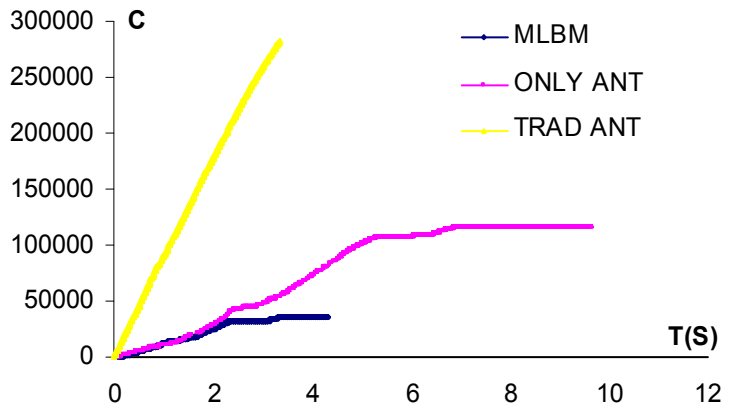
٤-٥-٨- مقایسه حجم ارتباطات مورچه ها در روشهای مختلف

معرفی

در این آزمایش حجم ارتباطات انجام شده (C) در مبنای زمان (T) در روش MLBM، روش مبتنی بر مورچه های هوشمند و روش مورچه های اولیه مورد مقایسه قرار گرفته اند. آزمایش با رسیدن به همگرایی نهایی خاتمه می یابد.

تحلیل و بررسی

همانطور که در شکل ٤-۱۳ دیده می شود، میزان ارتباطات مورچه ها در حالتی که از روش MLBM استفاده شده است بسیار کمتر از میزان ارتباطات در حالتی است که روش مبتنی بر مورچه های هوشمند به تنهایی بکار رود. در هر حال دیده می شود که در هر دو حالت میزان ارتباطات لازم کمتر از روش مبتنی بر مورچه های اولیه است. این مساله به این دلیل رخ می دهد که در روش جدید، در هر بار، مورچه S حرکت انجام داده و پس از آن k گره را بالانس می کند، در حالی که در روش اولیه هر مورچه $2S+1$ حرکت انجام داده و در انتها فقط دو گره را بالانس می کند. در شکل٤-١٤ مشخص است که در لحظات آخر، در روش جدید حجم ارتباطات تقریباً ثابت می شود. این امر به این دلیل رخ داده که در لحظات آخر، یعنی زمانی که سیستم به همگرایی نهایی نزدیک شده است، طول عمر مورچه زیاد و در نتیجه فرکانس موازنه بار کم می شود، بنابراین مدت بیشتری طول می کشد تا سیستم به همگرایی نهایی برسد. در ضمن به هر مورچه روش جدید $k=7$ واحد فضای حافظه اختصاص یافته است.



شکل ٤-۱۳. مقایسه حجم ارتباطات لازم برای نیل به همگرایی نهایی در روش MLBM، روش مبتنی بر دسته مورچه های هوشمند و روش مبتنی بر مورچه های اولیه.

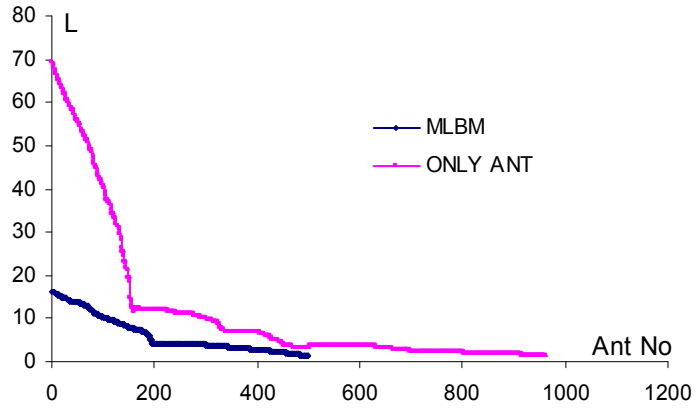
٤-٥-٩- بررسی تعداد مورچه ها نسبت به میزان همگرایی در روشهای مختلف

معرفی

همانطور که در قسمت قبل ادعا شد، یکی از ایرادات روش مبتنی بر دسته مورچه ها، تعداد زیاد مورچه ها و در نتیجه حجم زیاد ارتباطات لازم آنها می باشد. همچنین بیان شد که برای رفع این مشکل و قیاس پذیر کردن آن، در الگوریتم MLBM، روش مبتنی بر مورچه ها را با الگوریتم سطح همسایگی ترکیب کرده ایم. در این آزمایش، که در شکل ٤-١٤ نشان داده شده است، میزان کم شدن مورچه ها *(Ant No)* به همراه میزان همگرایی حاصل از آنها *(L)* در روش MLBM نشان داده شده است.

تحلیل و بررسی

همانطور که در شکل ٤-١٤ دیده می شود، تعداد مورچه های لازم در روش MLBM به میزان قابل توجهی به نسبت روشی که در آن الگوریتم مبتنی بر گروه مورچه های هوشمند به تنهایی به کار رفته، کاهش یافته است. مسلماً این امر به دلیل تاثیری است که روش سطح همسایگی بر روی بار گره ها می گذارد. به عنوان مثال در شکل ٤-١٤ دیده می شود که در همان ابتدای کار سطح موازنه بار سیستم به میزان قابل توجهی کاهش یافته است، بنابراین تعداد کمی مورچه به وجود آمده اند؛ این روال تا انتهای کار سیستم و رسیدن به همگرایی نهایی ادامه یافته است. بنابراین می توان ادعا کرد که ترکیب دو روش و ارائه روش MLBM، باعث قیاس پذیری بیشتر الگوریتم موازنه بار می شود.

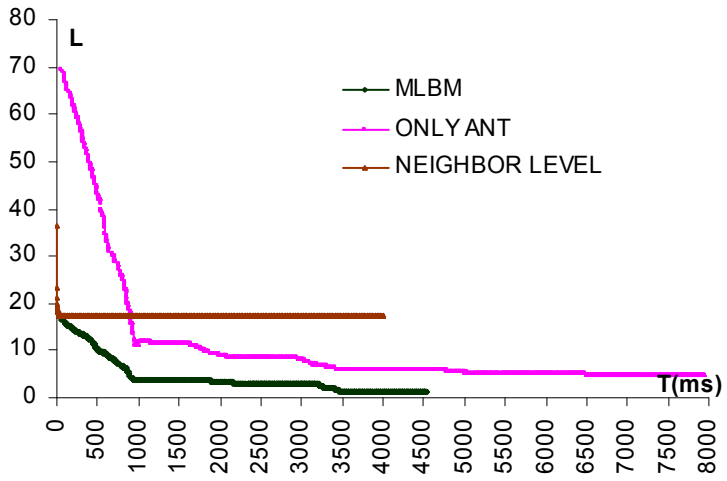


شکل ٤-١٤. مقایسه تعداد مورچه های به وجود آمده در روش MLBM با تعداد مورچه های به وجود آمده در روش مبتنی بر گروه مورچه های هوشمند.

٤-٥-١٠- بررسی سرعت همگرایی روشهای مختلف موازنه بار

معرفی

در این آزمایش سرعت همگرایی در روشهای مختلف با هم مقایسه شده اند. برای این منظور، میزان همگرایی *(L)* به وجود آمده از روشهای مختلف در زمانهای مختلف *(T)* مورد آزمایش قرار گرفتند.



شکل ٤-١٥. سرعت همگرایی روش MLBM، سطح همسایگی و مورچه های هوشمند.

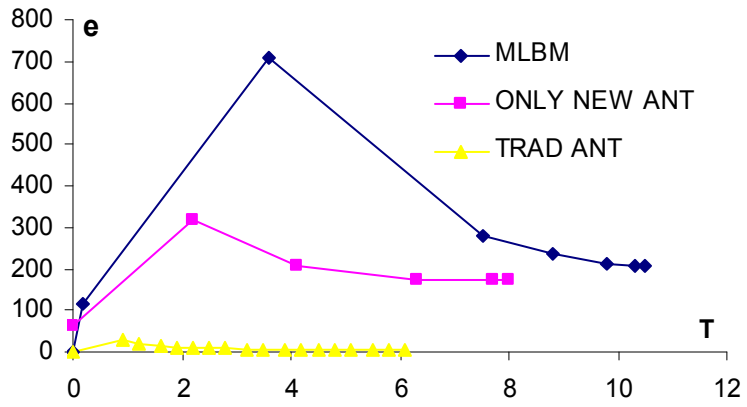
تحلیل و بررسی

همانطور که قبلاً بیان شد، ضعف روش سطح همسایگی در سرعت همگرایی پایین آن است. در شکل ٤-١٥ نشان داده ایم که استفاده ترکیبی این الگوریتم با الگوریتم مبتنی بر مورچه ها در MLBM باعث کم شدن زمان لازم برای فرایند موازنه بار است. به عبارت دیگر سرعت کند روش سطح همسایگی و خاصیت پخش ناحیه ای آن در ترکیب با روش مورچه ها از بین می رود.کم شدن زمان لازم برای MLBM، نسبت به هر دو الگوریتم تشکیل دهنده آن، نشان دهنده تعامل خوب دو الگوریتم می باشد.

٤-٥-١١- بررسی کارایی روش MLBM

معرفی

در آخرین آزمایش به مقایسه کارایی حاصل از روشهای مختلف می پردازیم. برای این منظور کارایی روش MLBM را با روش مبتنی بر مورچه های هوشمند و همچنین روش مبتنی بر مورچه های اولیه در ARMS مقایسه کرده ایم. میزان کارایی (e) که طبق رابطه ٤-٧ محاسبه می شود، در زمانهای مختلف (T) تا رسیدن به همگرایی نهایی با هم مقایسه شده اند. در این آزمایش میزان حافظه مورچه ها در روش MLBM و همچنین الگوریتم سطح همسایگی $K=۷$ در نظر گرفته شده است. همچنین در روش مبتنی بر مورچه های اولیه هم تعداد مورچه ها ۲۳۰ عدد در نظر گرفته شده اند، که در هر دوره ۱۵ گام برمی دارند.



شکل ٤-١٦. مقایسه کارایی روشهای مختلف ارائه شده.

تحلیل و بررسی

در دو آزمایش قبلی نشان داده شد که با استفاده از MLBM در تمام زمان اجرای فرایند موازنه بار، هم میزان ارتباطات *(C)* کم شده و هم میزان موازنه بار بیشتری حاصل شده است (افزایش L_k-L). بنا بر این آزمایشها و با توجه به رابطه ٤-۷ می توان نتیجه گرفت که میزان کارایی *(e)* نسبت به الگوریتم مبتنی بر گروه مورچه های هوشمند بیشتر است؛ که این مساله در شکل ٤-١٦ هم دیده می شود.

٤-٦- جمع بندی

در این فصل سعی شده تا مدلهای پیشنهادی برای موازنه بار به همراه جزئیات و نحوه پیاده سازی آنها در محیط ARMS شرح داده شوند. سپس محیط مورد استفاده برای انجام شبیه سازی تشریح شده است. در مرحله بعد با یک اثبات ریاضی برتری روش ارائه شده را به نسبت روش اولیه اثبات کرده ایم (این اثبات در قسمت بعدی توسط آزمایشهای انجام شده، تایید می گردند). در مرحله بعد شروع به انجام آزمایشهایی برای نشان دادن رفتار سیستم کرده ایم. سعی شده پارامترهایی مورد آزمایش قرار گیرند که به بهترین نحو بیان کننده رفتار سیستم ارائه شده باشند. این پارامترها در اکثر کارهای قبلی که بر روی موازنه بار انجام شده اند مورد استفاده قرار گرفته اند. آزمایشهای انجام شده بر روی الگوریتم های پیشنهادی به ویژه مکانیزم MLBM نشان می دهند که این الگوریتم به نسبت به هر یک از اجزای خود (که هر یک می توانند به عنوان روشی مستقل برای موازنه بار استفاده شوند) مقیاس پذیرتر بوده، کارایی بهتری را حاصل کرده و همچنین سرعت همگرایی بیشتری دارد. همچنین این مکانیزم به سرعت با شرایط محیطی منطبق می شود (انطباق پذیری)، کاملاً شفاف از دید کاربر، دارای سربار ارتباطی کم و همچنین عادلانه می باشد. با در نظر گرفتن شرایط و ویژگیهای منحصر به فرد مدل مدیریتی ARMS برای محیط گرید، ادعا می شود که مدل موازنه بار ارائه شده می تواند به عنوان مکانیزم موازنه بار ایده ال، تحت آن محیط مورد استفاده قرار گیرد.

فصل ۵

نتیجه گیری و کارهای آینده

۵-۱- نتیجه گیری

در تحقیق انجام گرفته در این پایان نامه، یک مدل موازنه بار برای محیط محاسباتی گرید ارائه شده است. مدل ارائه شده که MLBM نام دارد، تحت سیستم مدیریت منابع عامل گرای ARMS عمل می کند. درMLBM سعی شده تا حد امکان، نیاز ها و ویژگیهای لازم برای یک سیستم موازنه بار، همچون قیاس پذیری، پایداری، تطبیق پذیری، شفافیت از دید کاربر و حداقل بودن سربار ارتباطی برآورده شوند. یکی از مسائل مهم در مکانیزم های موازنه بار، نحوه اندازه گیری بار گره ها می باشد. روشهای موجود موازنه بار از مقیاس های مختلفی، همچون تعداد پردازشهای موجود در هر گره، نرخ ورود تقاضاها به هر گره، متوسط زمان پاسخ پردازشها برای اندازه گیری بار گره ها استفاده می کنند. با توجه به اهمیت این فاکتور در مکانیزم موازنه بار، درMLBM سعی شده روش جدیدی برای بیان میزان بار یک گره ارائه شود. این مقیاس وابسته به قسمتی از ساختار سیستم مدیریت منابعARMS به نام PACE می باشد. عمل موازنه بار نیاز به جستجو برای یافتن گره های پربار و کم بار دارد. چنین مساله ای را در محیط گرید را می توان به عمل جستجو در یک گراف تشبیه کرد. در چنین مسائلی روشهای اکتشافی، همانند روشهای مبتنی بر دسته مورچه ها انتخاب مناسبی برای حل مسئله هستند. در تحقیق انجام گرفته در این پایان نامه نیز علاوه بر اینکه کاربرد این روش در موازنه بار گرید اثبات شده، سعی در رفع بعضی اشکالات و ضعفهای ذاتی آن نیز گردیده است.

به طور کلی MLBM، یک مدل موازنه بار چند سطحی است. این مکانیزم عملکرد خود را در سه سطح مجزا انجام می دهد. علی رغم اینکه این سطوح مجزا از هم هستند ولی نتیجه کار هر یک از آنها بر روی کارایی دیگری تاثیر مستقیم دارد.

سطح اول (سطح محلی)، دامنه عملکردش در درون یک گره گرید می باشد. به عبارت دیگر در این سطح تلاش می شود بار کاری محول شده به یک گره به نحوی بر روی پردازنده های آن گره زمانبندی شود که هم منابع بیشترین کارایی را داشته باشند و هم تمام پردازشها بتوانند قبل از موعد زمانی مقرر خود تمام شوند. این الگوریتم چند هدفه، با استفاده از الگوریتم های ژنتیک در ARMS پیاده سازی شده است.

در سطح دوم MLBM (سطح همسایگی)، الگوریتم موازنه بار سطح همسایگی قرار دارد. این الگوریتم جزء روشهای موازنه بار در سطح زمانبند می باشد. یعنی عملکرد آن در درون زمانبند هر گره است. این گونه الگوریتمهای موازنه بار معمولاً سربار کمی را به سیستم تحمیل می کنند. در این الگوریتم، هر گره به صورت دوره ای شروع به جمع آوری اطلاعات بار کاری همسایه های خود می کند. اگر گره نسبت به همسایه هایش پربار باشد، برای همسان ساختن بار خود با همسایه ها، زیر مجموعه ای از همسایه ها که بار آنها کمتر از میانگین بوده را انتخاب کرده و سعی در ارسال اضافه بار خود به آنها یی می کند که کمترین هزینه ارسال را دارند. استفاده از این الگوریتم، باعث پخش ناحیه ای بار می گردد. نحوه پخش بار در این روش را می توان به یک حرکت موجی شکل تشبیه کرد که دامنه آن با سرعت کمی افزایش می یابد. این مساله عیب اصلی این روش می باشد. عیب دیگر این روش دانش محدود نسبت به کل سیستم و در نتیجه عدم توانایی در تصمیم گیری بهینه است. اما در کنار این عیوب، عادلانگی اجرای موازنه بار، کم بودن سربار ارتباطی و همچنین در نظر گرفتن هزینه ارسال بار از نکات قوت این روش محسوب می شوند. باید توجه داشت که این الگوریتم را می توان به تنهایی نیز به عنوان یک الگوریتم موازنه بار مورد استفاده قرار داد.

در سطح سوم MLBM (سطح گرید)، الگوریتمی با استفاده از الگوی کاری مورچه ها برای موازنه بار ارائه شده است. در این الگوریتم اکوسیستمی از مورچه ها که وظیفه موازنه بار را دارند ایجاد شده است. منظور از اکوسیستم این است که در این سیستم جمعیتی از مورچه ها بنا به نیاز به وجود آمده و در شرایطی هم که احتیاجی به آنها نیست از بین می

روند. مورچه های به وجود آمده در طی دوره حیات خود عملیات موازنه بار را در طی دوره های مختلفی انجام می دهند. زمانی که مورچه ها احساس کنند که سیستم از شرایط موازنه بار فاصله زیادی دارد شروع به تولید مثل می کنند و بالعکس در شرایطی که سیستم را نزدیک به حالت موازنه درک کنند، دست به خودکشی می زنند. این نحوه عملکرد باعث ایجاد تطبیق پذیری با شرایط محیطی می شود. به عبارت دیگر تعداد مورچه ها بنا به سطح اختلاف بار موجود در سیستم تغییر می کند. تعداد گامهای مورچه نیز که یکی دیگر از پارامترهای موثر در کارایی الگوریتم می باشد در هر دوره کاری مورچه متغیر است. به بیان دیگر، تعداد گامهای مورچه مشخص کننده فرکانس انجام موازنه بار در سیستم است. اگر تعداد گامهای مورچه کم باشد، تعداد دفعات انجام فرایند موازنه بار افزایش می یابد، بالعکس تعداد گامهای زیاد مورچه، باعث کم شدن تعداد دفعات رخ دادن موازنه بار در سیستم می شود. مسلماً حالت اول در شرایط نزدیک بودن به حالت موازنه و حالت دوم برای شرایط دور از حالت موازنه مناسب است. بنا بر این اصل است که در سیستم ارائه شده تعداد گامهای مورچه در دوره های مختلف زندگی اش متفاوت انتخاب شده است. تعداد متغیر مورچه ها و همچنین متغیر بودن تعداد گامهای آنها تطبیق پذیری خوب و قابل قبولی را به وجود می آورد. یکی از اشکالات مطرح در مدلهای مبتنی بر دسته مورچه ها، سربار ارتباطی زیاد آنهاست. برای رفع این اشکال در اکوسیستم ارائه شده به مورچه ها حافظه داده شده است. حافظه داده شده به مورچه ها سبب افزایش کارایی آنها در عملیات موازنه بار می شود. برای این منظور حافظه داده شده به مورچه ها به دو قسمت تقسیم می شود؛ در یک قسمت مشخصات گره های کم باری که مورچه در حین حرکت خود دیده است و در قسمت دیگر مشخصات گره های پربار دیده شده قرار دارد. البته این حافظه زمانی باعث افزایش کارایی می شود که مورچه بتواند آن را در طی دوره زندگی خود پر کند. با توجه به اینکه بسیاری از گره هایی که مورچه ملاقات می کند ممکن است در حالت موازنه باشد (نه پربار باشند و نه کم بار)، این احتمال وجود دارد که این حافظه خالی بماند. برای رفع این مشکل در این الگوریتم از راهکار ابتکاری "موازنه بار در سطح مورچه ها " استفاده کرده ایم. این راهکار از این مساله که ممکن است در شرایطی چندین مورچه به طور همزمان در یک گره باشند استفاده کرده است. در این شرایط مورچه های هم مکان می توانند اطلاعاتی را که تا کنون به دست آورده اند با هم تقسیم کنند. این عمل نوعی موازنه اطلاعات در سطح مورچه ها است.

با توجه به آنچه گفته شد الگوریتم مورچه ها به این شکل عمل می کند که هر گره در شرایطی که خود را پربار حس کند، شروع به ایجاد مورچه و رها کردن آن در محیط می کند. این گره اطلاعات خود را به عنوان یک گره پربار در حافظه مورچه قرار می دهد. مورچه پس از طی تعداد گام مشخصی اطلاعات گره های پربار و کم بار بدست آمده را با هم موازنه می کند. پس از این مرحله، مورچه با استفاده از دانشی که در دوره اخیر از محیط به دست آورده است تعداد گامهای خود را برای دوره بعدی مشخص می کند و به این ترتیب یک دوره کاری دیگر برای مورچه آغاز می شود. در نهایت ممکن است تعداد گامهای مشخص شده برای دوره بعدی مورچه آنقدر زیاد باشد که عملاً حضور مورچه در سیستم فایده ای نداشته باشد. در چنین شرایطی مورچه تصمیم به خود کشی می گیرد. این بدان معناست که مورچه هیچ دوره جدیدی را شروع نخواهد کرد.

دسته هایی از این مورچه های هوشمند می توانند موازنه بار خوبی را در سیستم به وجود آورند. این روش به عنوان سطح سوم مکانیزم MLBM مورد استفاده قرار گرفته است. این روش ضمن اینکه به تنهایی نیز به عنوان یک روش موازنه بار مطرح است سعی در موازنه بار در سطح وسیعی از گرید می کند. به عکس روش قبلی این روش بسیار سریع به همگرا می شود. عادلانه نبودن و حجم سربار ارتباطی زیاد دو عیب عمدهٔ این روش می باشند.

همانطور که گفته شد، مکانیزم MLBM یک روش چند سطحی است. تعامل روش سطح همسایگی و روش مورچه ها در این سیستم سبب از بین رفتن معایب آنها و در نتیجه انجام عمل موازنه بار با کارایی بالا می گردد. در اینصورت استفاده از الگوریتم سطح همسایگی و موازنه بار حاصل از آن سبب می شود که به تعداد کمی مورچه نیاز باشد. در طرف دیگر استفاده از الگوریتم مورچه ها می تواند باعث افزایش سرعت همگرایی روش سطح همسایگی و همچنین برطرف ساختن ضعف آن در اتخاذ تصمیمات نادرست شود. عملکرد الگوریتم MLBM به این صورت است که هر گره گرید با استفاده از روش سطح همسایگی به صورت دوره ای بارش را با همسایه هایش تقسیم می کند. اگر گرهی پس از چندین دوره موازنه بار هنوز پر بار بود، یک مورچه ایجاد کرده و آنرا در محیط رها می کند تا بار خود را در دامنه وسیعتری از سطح همسایگی پخش کند. ممکن است یک گره خود مورچه ای تولید نکند، اما مورچه های دیگر از آن عبور کنند و اطلاعات آنرا در خود ثبت کنند و بعداً آنرا مورد استفاده قرار دهند. با پیگیری این طرز عملکرد سیستم با سرعت زیادی همگرا می شود. چنین سیستمی قیاس پذیر، تطبیق پذیر و شفاف از دید کاربر می باشد.

به علت جدید بودن مبحث گرید، تحقیقات، آزمایشها و بررسی های انجام شده بر روی ایده های ارائه شده در موازنه بار گرید، هنوز در مراحل مقدماتی است و نیاز به کار و تحقیق بیشتر محققان بر روی آن را طلب می کند، لیکن با تکیه بر همین نتایجی که تا کنون به دست آمده نیز می توان ادعا کرد که این روشها می توانند تا حد زیادی موانع موازنه بار در محیط پیچیده گرید را با کیفیتی بالا، بدون فشار آوردن به زیر ساخت و یا دخالت دادن کاربر در آن، از سر راه بردارد. البته در ابتدای راهی طولانی هستیم و کارهای زیادی برای عملیاتی شدن این طرحها باید انجام شود. در بخش بعدی به بعضی از این موارد اشاره می کنیم.

۵-۲-کارهای آینده

در ادامه کارهای انجام گرفته در این پایان نامه و با توجه به باز بودن مبحث مدیریت منابع در گرید و همچنین اهمیت موازنه بار در آن کارهای زیادی می تواند انجام شود که در زیر به چند مورد از آنها اشاره می شود.

در این پایان نامه سعی شده همه ایده ها و نتایج مورد انتظار آنها به صورت شبیه سازی پیاده سازی شوند. با توجه به منطقی بودن روش و همچنین مجزا بودن قسمتهای مختلف آن از هم، به نظر می آید ارائه یک مدل ریاضی برای آن گام بلندی در جهت عملیاتی شدن و مستحکم تر شدن پایه های تئوریک آن می باشد.

همانطور که در پایان نامه اشاره شد، یکی از ویژگیهای مهم الگوریتم موازنه بار تطبیق پذیری آن با شرایط محیطی می باشد. در این راه، تعداد مورچه ها و تعداد گامهای آنها را به صورت تطبیق پذیر ایجاد کردیم. در ادامه این رویکرد و در جهت افزایش تطبیق پذیری و بهره وری، می توان میزان حافظه مورچه ها را نیز به صورت تطبیق پذیری تعریف کرد. در اینصورت می توان در سیستم مورچه های چاق و لاغر داشته باشیم که هر یک در اثر بروز شرایط خاصی در سیستم ایجاد می شود. پیش بینی می شود که این مساله تاثیر به سزایی در کاهش سربار ارتباطی داشته باشد.

گرید محیط رو به گسترشی است که البته هنوز تا حد زیادی در مراحل تحقیقاتی به سر می برد. چون گرید در سطح جهان گسترده خواهد شد و گره های مختلف آن متعلق به سازمانهای مختلفی با سیاست گذاریهای متفاوت خواهند بود و اغلب آنها نیز اهداف تجاری را دنبال می کنند، طیف گسترده ای از تحقیقات در گرید را مسائل مربوط به قراردادهای تجاری آن تشکیل می دهد. از این مساله در گرید تحت عنوان فروش قدرت محاسبات یاد می شود. این مساله در بحث کشف و تبلیغ سرویس و همچنین موازنه بار بسیار حائز اهمیت است. در ادامه کار این پایان نامه می توان مدل ارائه شده

را به گونه ای قدرتمند ساخت که بتواند ضمن برقراری موازنه بار، قراردادهای تجاری بین سازمانهای مجازی مختلف را نیز رعایت کند.

یکی از مسائل مهم دیگر در محیط گرید برقراری امنیت برای پردازشها و داده های لازم آنها می باشد. مساله امنیت در محیط گرید بسیار مهمتر از مسائل امنیتی در اینترنت امروزی است و پیچیده گیهای خاص خود را دارد. این مسئله سیستمهای مدیریت منابع گرید را با چالشی جدی مواجه ساخته است. در ادامه این پایان نامه می توان از روالهای موجود برای امن ساختن محیط ARMS استفاده کرد و در مرحله بعد MLBM را با آن منطبق ساخت.

یکی دیگر از کار هایی که در ادامه این پایان نامه می توان انجام داد، پیاده سازی MLBM بر روی سیستمهای مدیریت منابع معروف گرید همچون گلوباس و کندر است.

همانطور که در فصل ٤ دیده شد در الگوریتم سطح همسایگی، گره فرستنده همواره سعی می کند کارها را به سمت گره هایی ارسال کند که میزان بار آنها به اندازه θ از میانگین بدست آمده کمتر است. میزان θ بر روی میزان سودمندی ارسال تاثیر به سزایی دارد. این مساله در آزمایشهای انجام شده نشان داده شده است. در مدل ارائه شده، مقدار θ ثابت در نظر گرفته شده است. در ادامه می توان میزان این پارامتر را نیز به صورت تطبیقی و بنا به شرایط سیستم تعریف کرد.

در فصل ٤ ایده ای برای اندازه گیری دقیق بار ارائه شد، این نحوه اندازه گیری بار، هنوز احتیاج به کار بیشتر دارد. به ویژه تعیین الگوریتمی برای بدست آوردن میزان $\overline{T}$ (میزان زمان متوسط اجرای برنامه) جزء کارهای آینده این پایان نامه محسوب می شود.

مراجع

[١]. A. Grimshaw, W. Wulf et al. The Legion Vision of a Worldwide Virtual Computer, *Communications of the ACM, vol. ٤٠(١). ١٩٩٧.*

[٢]. A. Montresor, H. Meling, and Ö. Babaoglu, "Messor: Load-Balancing through a Swarm of Autonomous Agents", in Proc. of ١st Int. Workshop on Agents and Peer-to-Peer Computing, ١st ACM Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems, Bologna, Italy, ٢٠٠٢.

[٣]. A. Shaout and P. McAuliffe, Job scheduling using fuzzy load balancing in distributed system, *in Proc. of ٦st conf ICPAD, ١٩٩٨.*

[٤]. A. Y. Zomaya and Y. Teh, Observations on using genetic algorithms for dynamic load-balancing, *IEEE Trans. on Parallel and Distributed Systems* ٩ (September ٢٠٠١), pp.٨٩٩-٩١١.

[٥]. B. Segal, "Grid Computing: The European Data Grid Project", *in Proc. of IEEE Nuclear Science Symp. And Medical Imaging Conf., Lyon, France, ٢٠٠٠.*

[٦]. B. S. Joshi, Seyed H. Hosseini, A Methodology for Evaluating Load Balancing Algorithms, *in Proc. Of ٦th Int. Conf. IEEE HPDC, ١٩٩٣*

[٧]. B.Veeravalli and W. Han Min, Scheduling Divisible Loads on Heterogeneous Linear Daisy Chain Networks with Arbitrary Processor Release Times, *IEEE TRANS. ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. ١٥(٣), MARCH ٢٠٠٤*

[٨]. D. Gedye and C. Kasnoff *Seti@Home* ٢٠٠٢,http://setiathome.berkeley.edu

[٩]. D. E. Atkins, W. P. Birmingham, Toward Inquiry-Based Education Through Interacting Software Agents, *IEEE Computer, Vol. ٢٩, No. ٥, pp. ٦٩-٧٦, ١٩٩٦.*

[١٠]. D. Giddy, J. Kotler, High Performance Parametric Modeling with Nimrod/G: Killer Application for the global Grid, *١٤th Int. Parallel and Distributed Processing Symposium, April ٢٠٠٠.*

[١١]. D.Grosu, T. Chronopoulos,Algorithmic Mechanism Design for Load Balancing in Distributed Systems, *IEEE TRANS. ON SYSTEMS, MAN, AND CYBERNETICS-PART B: CYBERNETICS, VOL. ٣٤(١), ٢٠٠٤.*

[١٢]. D. Grosu, A. T. Chronopoulos, and M. Y. Leung, Load balancing in distributed systems: An approach using cooperative games, *Proc. ١٦th IEEE Int. Parallel Distributed Processing Symp., pp. ٥٢–٦١،٢٠٠٢.*

[١٣]. Distributed.net ٢٠٠٢ http://distributed.net/

[١٤]. D.L. Eager, E.D. Lazowska, and j. Zahorjan, Adaptive load sharing in homogeneous distributed systems, *IEEE Trans. Software Eng.*, vol. ١٢(٥), pp. ٦٦٢-٦٧٥, ١٩٨٦.

[١٥]. D. Gerbec, S. Gasperic, I. $Smon and F. Gubina, Determining the load profiles of consumers based on fuzzy logic and probability neural networks, *IEEE Proc.-Gener. Transm. Distrib., Vol. ١٥١(٣), ٢٠٠٤.*

[١٦]. E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Arti_cial Systems. Oxford University Press, ١٩٩٩.*

[١٧]. F. Bonomi and A. Kumar, Adaptive Optimal Load-Balancing in a Heterogeneous Multiserver System with a Central Job Scheduler, *IEEE Trans. Computers*, *vol. ٣٩(١٠), pp. ١٢٣٢-١٢٥٠،١٩٩٠*.

[١٨]. F. Kon, R. Campbell, M. Mickunas, and K. Nahrstedt, ٢K: A Distributed Operating System for Dynamic Heterogeneous Environments, *in Proc. of ٩th IEEE Int. Symp. on High Performance Distributed Computing, ٢٠٠٠*.

[١٩]. F. Berman, G. Fox, and T. Hey, eds, Grid Computing: Making the Global Infrastructure a Reality,. *John Wiley and Sons, ٢٠٠٣*.

[٢٠]. I. Foster and C. Kesselman, eds.Grid: Blueprint for a New Computing Infrastructure, *Morgan Kaufman, ١٩٩٩, ٢٧٩-٣٠٩*.

[٢١]. Gridstart web site. ٢٠٠٢. http://www.gridstart.org

[٢٢]. *Globus project web site* University of Chicago ٢٠٠٢, http://www.globus.org

[٢٣]. H.C. Lin and C.S. Raghavendra, A Dynamic Load-Balancing Policy with a Central Job Dispatcher (LBC*), IEEE Trans. Software Eng., vol. ١٨, no(٢), pp. ١٤٨-١٥٨, ١٩٩٢*.

[٢٤]. H. Lieberman, "Autonomous Interface Agents", *in CHI '٩٧ Conf. Proc. on Human Factors in Computing Systems, pp. ٦٧-٧٤, ١٩٩٧*.

[٢٥]. H. Kameda, J. Li, C. Kim, and Y. Zhang, *Optimal Load Balancing in Distributed Computer Systems*. London, U.K.: Springer-Verlag, ١٩٩٧.

[٢٦]. H. S. Nwana, J. Rosenschein, Agent-Mediated Electronic Commerce: Issues, Challenges and Some Viewpoints, *in Proc. of ٢nd ACM Int. Conf. on Autonomous Agents, pp. ١٨٩-١٩٦, ١٩٩٨*.

[٢٧]. http://www.ccl-nece.de/macj/software.htm

[٢٨]. I. Foster, and C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, *Morgan-Kaufmann, ١٩٩٨*.

[٢٩]. I. Foster, C. Kesselman, and S. Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations, *in Intl. J. Supercomputer Applications, ٢٠٠١*.

[٣٠]. I. Foster. Internet Computing and the Emerging Grid, *Nature*.December. ٢٠٠٠ http://www.nature.com/nature/webmatters/grid/grid.html

[٣١]. I. Foster, C. Kesselman, The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets. *J. Network and Computer Applications, ٢٠٠١*

[٣٢]. I. Foster, C. Kesselman, A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation, *A Report of a Workshop on Middleware, IETF, RFC ٢٧٦٨, ٢٠٠٠*. http://www.ietf.org/rfc/rfc٢٧٦٨.txt

[٣٣]. J. Cao, "Agent-Based Resource Management System (ARMS)," *PhD Thesis, Warwick University Dept. of Computer Science,٢٠٠١*.

[٣٤]. J. Case, M. Fedor, M. Schoffstall, and J. Davin, A Simple Network Management Protocol, *RFC ١٠٦٧, IETF Draft Standard, ١٩٨٨*.

[٣٥]. J. Cao, D. J. Kerbyson, and G. R. Nudd, Use of Agent-Based Service Discovery for Resource Management in Metacomputing Environment, *in Proc. of ٧th Int. Euro-Par Conf., Manchester, UK, Lecture Notes in Computer Science ٢١٥٠, Springer-Verlag, pp. ٨٨٢-٨٨٦, ٢٠٠١*.

[٣٦]. J. Cao, D. J. Kerbyson, E. Papaefstathiou, and G. R. Nudd, Performance modelling of parallel and distributed computing using PACE, *in Proc. ١٩th IEEE Int. Performance Computing and Communication Conference, pp. ٤٨٥-٤٩٢, Phoenix, AZ, USA, ٢٠٠٠.*

[٣٧]. J. Cao, D. J. Kerbyson, and G. R. Nudd, Dynamic Application Integration Using Agent-Based Operational Administration, *in Proc. of ٥th Int. Conf. on Practical Application of Intelligent Agents and Multi-Agent Technology, Manchester, UK, pp. ٣٩٣-٣٩٦, ٢٠٠٠.*

[٣٨]. J. Cao, Self-Organizing Agents for Grid Load Balancing, *Proc. of the Fifth IEEE/ACM Int. Workshop on Grid Computing (GRID'٠٤).*

[٣٩]. J. Cao, F. Zimmermann, Queue scheduling & advanced reservation(COSY)- *Proc. of the Fifth IEEE/ACM Int. Conference on Grid and Pervasive Computing.*

[٤٠]. J.Cao, Daniel P. Spooner, Agent-Based Grid Load Balancing Using Performance-Driven Task Scheduling, *In Proc. of ١٧th IEEE Int. Parallel & Distributed Processing Symposium (IPDPS ٢٠٠٣), Nice, France, April ٢٠٠٣.*

[٤١]. J. Cao, D. J. Kerbyson, E. Papaefstathiou, and G. R. Nudd, Performance Modelling of Parallel and Distributed Computing Using PACE, *in Proc. of ١٩th IEEE Int. Performance, Computing and Communication Conf., Phoenix, USA, pp. ٤٨٥-٤٩٢, ٢٠٠٠.*

[٤٢]. J. Liu, , X. Jin, and Y. Wang, Agent-Based Load Balancing on Homogeneous Minigrids: Macroscopic Modeling and Characterization, *IEEE TRANS.IONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. ١٦(١), ٢٠٠٥.*

[٤٣]. J. L. Deneubourg, S. Goss, N. Franks, The dynamics of collective sorting robot-like ants and ant-like robots. In *From Animals to Animates, Proc. of the First Int. Conference on Simulation of Adaptive Behavior, pages ٣٥٦-٣٦٣. MIT Press, ١٩٩٠*

[٤٤]. J. M. Bradshaw, ed., Software Agents, The AAAI Press /The MIT Press, ١٩٩٧.

[٤٥]. Jini.org. A central place and resource for the Jini CommunitySM, http://jini.org

[٤٦]. K.Mang, W.hong, Ant Colony Optimization for Routing and Load-Balancing Survey and New Directions, *IEEE TRANS.IONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, VOL. ٣٣, NO. ٥, SEPTEMBER ٢٠٠٣.*

[٤٧]. K. Krauter, R. Buyya, and M. Maheswaran, A Taxonomy and Survey of Grid Resource Management Systems, *Technical Report, Monash University, Australia, ٢٠٠٠.*

[٤٨]. K.Ming V. Yu, and C. Chou, A Fuzzy-Based Dynamic Load-Balancing Algorithm, *Proc. Of fuzzy System Application Symp. IEEE Press ٢٠٠٠*

[٤٩]. L. Anand, D. Ghose, and V. Mani, ELISA: An Estimated Load Information Scheduling Algorithm for Distributed Computing System, *Computers and Mathematics with Applications*, ٣٧, pp. ٥٧-٨٥, ١٩٩٩.

[٥٠]. L. Boloni, and D. Marinescu, An Object-Oriented Framework for Building Collaborative Network Agents, *in A. Kandel et al, eds, Agents in Intelligent Systems and Interfaces, Kluewer, ١٩٩٩.*

[٥١]. Chapin, S., Karpovich, J., Grimshaw, A., The Legion Resource Management System, *Proceedings of the ٥th Workshop on Job Scheduling Strategies for Parallel Processing, April ١٩٩٩.*

[٥٢]. L. Ferreira and V. Berstis. "Fundamentals of Grid Computing" *IBM Redbooks. IBM Publications Center. ٢٠٠٢.*

[٥٣]. L.SUN CHEUNG, A fuzzy approach to load balancing in a distributed object computing network, *In Proc. Of ٦th Int IEEE Conf. HPDC, ٢٠٠٠.*

[٥٤]. M. Baker, R. Buyya, and D. Laforenza, The Grid: A Survey on Global Efforts in Grid Computing, *Tech. Rep: ٢٠٠١/٩٢, Monash University, Australia, ٢٠٠١.*

[٥٥]. M. Dahlin, Interpreting Stale Load Information, *IEEE TRANS.IONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. ١١(١٠), ٢٠٠٠.*

[٥٦]. M. Dorigo. Optimization, learning and natural algorithms (in Italian). *doctoral dissertation, Politecnico di Milano, Dipartimento di Elettronica, Italy (١٩٩٢).*

[٥٧]. M. J. Wooldridge, and N. R. Jennings, Software Engineering with Agents: Pitfalls and Pratfalls, *IEEE Internet Computing, Vol. ٣(٣), pp. ٢٠-٢٧, ١٩٩٩.*

[٥٨]. M.Livny and M.Melman, Load balancing in homogenous broad cast distributed systems, *in proc. Conf. Performance, ACM, ١٩٨٢, pp. ٤٧-٥٥.*

[٥٩]. M. Litzkow, M. Livny, M. Mutka, Condor - A Hunter of Idle Workstations, *Proc.dings of the ٨th Int. Conference of Distributed Computing Systems, ١٩٨٨.*

[٦٠]. M. Mitzenmacher, how useful is old information, *IEEE Trans. vol ١١ (١), ٢٠٠٠.*

[٦١]. M. Resnick. Turtles, Termites, and Traf_c Jams: Explorations in Massively Parallel Microworlds. *MIT Press, ١٩٩٤.*

[٦٢]. M. Stonebraker, R. Devine, An Economic Paradigm for Query Processing and Data Migration in Mariposa, *٣rd Int. Conference on Parallel and Distributed InformationSystems*, ١٩٩٤. http://mariposa.cs.berkeley.edu:٨٠٠٠/mariposa/

[٦٣]. Northeast Parallel Architectures Center, Syracuse University RSA Factoring-By-Web project ١٩٩٦, http://www.npac.syr.edu/factoring.html

[٦٤]. N. R. Jennings, and M. J. Wooldridge (eds), Agent Technology: Foundations, Applications, and Markets, *Springer-Verlag, ١٩٩٨.*

[٦٥]. O. F. Rana, and D. W. Walker, The Agent Grid: Agent-Based Resource Integration in PSEs, in *Proc. of ١٦th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation, Lausanne, Switzerland, ٢٠٠٠.*

[٦٦]. O. Krem and J. Kramer, Methodical Analysis of Adaptive Load Sharing Algorithms, *IEEE Trans. On High Performance Computing Vol ٤(٣), ١٩٩٢.*

[٦٧]. Ö. Babaoglu, H. Meling, and A. Montresor, Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems, *in Proc. of ٢٢nd IEEE Int. Conf. on Distributed Computing Systems, Vienna, Austria, pp. ١٥-٢٢, ٢٠٠٢.*

[٦٨]. Oram , Peer-to-Peer: Harnessing the Power of Disruptive Technologies, *O'Reilly Press,USA, ٢٠٠١*

[٦٩]. P. Birman and T. A. Joseph, Exploiting virtual synchrony in distributed systems, *in Proc. ACM-SIGOPS ١١th Symp. Oper. Syst. Principles, ١٩٨٧, pp. ١٢٣-١٣٨.*

[٧٠]. P. Ciancarini, and M. Wooldridge (eds.), Agent-Oriented Software Engineering, *Lecture Notes in Artificial Intelligence, Vol. ١٩٥٧, Springer Verlag, ٢٠٠١.*

[٧١]. R. A. Brooks and P. Maes, actions to global tasks: Stigmergy and collective robotics.In, Proc. *of the Fourth Int. Workshop on the Synthesis and Simulation of Living Systems, pp. ١٨١-١٨٩, ١٩٩٤.*

[٧٢]. R. Buyya, David Abramson and Jon Giddy, Nimrod/G: Architecture for a Resource Management and Scheduling System in a Global Computational Grid, *٤th Intl. Conf. on High Performance Computing in Asia-Pacific Region (HPC Asia ٢٠٠٠), China*.

[٧٣]. R. Buyya, S. Chapin, and D. DiNucci, Architectural Models for Resource Management in the Grid**,** *In Proc of ٨th Int Symp on High Performance Computing, ١٩٩٩*

[٧٤]. R. Buyya ,High Performance Cluster Computing, *Prentice Hall, ١٩٩٩*.

[٧٥]. R. G. Davison, J. J. Hardwicke, and M. D. J. Cox, Applying the Agent Paradigm to Network Management, *Proc. ١٩٩٩ ACM Symp.On Applied Computing ١٩٩٩, pp. ٤١٠-٤١٦*.

[٧٦]. R. J. Kuo, C.Y. Chiu, Integration of Fuzzy Theory and Ant Algorithm for Vehicle Routing Problem with Time Window, *In Proc. ٩th IEEE Symposium on NCS, ٢٠٠٠, IEEE Press*

[٧٧]. R. Mirchandaney, D.Towsleym, and J. A Stankovic, Analysis of the effects delays on load sharing*, IEEE Trans. Comput., vol. C-٣٨(١١), pp. ١٥١٣-١٥٢٥, ١٩٨٩*.

[٧٨]. Realizing the Information Future: The Internet and Beyond. *National Academy Press, ١٩٩٤*. http://www.nap.edu/readingroom/books/rtif/.

[٧٩]. R. Stevens, P. Woodward, From the I-WAY to the National Technology Grid. *Communications of the ACM, ٤٠(١١), pp. ٥٠-٦١. ١٩٩٧*.

[٨٠]. S**.** Dhaka١, B. S**.** Paskaleva, Dynamical Discrete-Time Load Balancing in Distributed Systems in the presence of Time Delays, *in Proc. ٤٢nd IEEE Decision and Control Conference , pp.٥١٢٨- ٥١٣٤ Vol.٥, ٢٠٠٣*.

[٨١]. S. Lalis, A. Karipidis, JaWS: An Open Market-Based Framework for Distributed Computing over the Internet, *IEEE/ACM Int. Workshop on Grid Computing (GRID ٢٠٠٠)*, Dec. ٢٠٠٠. http://roadrunner.ics.forth.gr:٨٠٨٠

[٨٢]. S.Viswanathan, B. Veeravalli, Design and Analysis of a Dynamic Scheduling Strategy with Resource Estimation for Large-Scale Grid Systems. *In Proc. ٥th IEEE/ACM Int. Grid Computing Workshop, pp.١٦٣-١٧٠, ٢٠٠٤*.

[٨٣]. S. Zhou, A trace-driven simulation study of dynamic load balancing, IEEE Trans. Software Eng, vol. ١٤(٩), pp.١٣٢٧-١٣٤١, ١٩٨٨.

[٨٤]. T. W. Malone, and K. Crowston, The Interdisciplinary Study of Coordination, *ACM Computing Survey, Vol. ٢٦(١), pp. ٨٧-١١٩, ١٩٩٤*.

[٨٥]. T. L. Casavants and J. G. Kuhl, A taxonomy of scheduling in general-purpose distributed computing systems, *IEEE Trans. Software Eng., vol. SE-١٤(٢), pp. ١٤١-١٥٤, ١٩٨٨*.

[٨٦]. W.Hoschek, J. Jaen-Martinez, Data Management in an Int. Data Grid Project, *Proc. of the first IEEE/ACM Int. Workshop on Grid Computing, (Springer Verlag Press, Germany), India, ٢٠٠٠*

[٨٧]. W. Leinberger, and V. Kumar, Information Power Grid: The New Frontier in Parallel Computing, *IEEE Concurrency, Vol. ٧(٤), pp. ٧٥-٨٤, ١٩٩٩*.

[٨٨]. W. Yeong, T. Howes, and S. Kille, Lightweight Directory Access Protocol, *RFC ١٧٧٧, IETF Draft Standard, ١٩٩٥*.

[٨٩]. Y. Fan, and J. Cao, Multi-Agent Systems: Theories, Applications and Methods, Tsinghua *University Press / Springer-Verlag, ٢٠٠١*.

[۹۰]. Y. Lan and T. Yu, A Dynamic Central Scheduler Load-Balancing Mechanism, *Proc. IEEE ۱٤th Ann. Int'l Phoenix Conf. Computers and Comm., pp. ۷۳٤-۷٤۰, ۱۹۹٥.*

[۹۱]. Y. Zhu, Yiming Hu, Efficient, Proximity-Aware Load Balancing for DHT-Based P۲P Systems, *Proc. of the Third Int. Conference on Peer-to-Peer Computing (P۲P'۰۳).*

[۹۲]. Z. Zeng and B. Veeravalli, Rate-Based and Queue-Based Dynamic Load Balancing Algorithms in Distributed Systems, *۱۰th Int. Conference on Parallel and Distributed Systems ,IEEE ۲۰۰۰.*

[۹۳]. Z. Zeng and V. Bharadwaj, A Static Load Balancing Algorithm via Virtual Routing, *Parallel and Distributed Computing and Systems, Marina del Rey, CA, USA, pp. ۲٤٤-۲٤۹, ۲۰۰۳.*

[۹٤]. ابراهیم باقری، مدل معرفی و اکتشاف منابع مبتنی بر عاملهای سیار برای محیطهای محاسباتی فراگیر رادما، پایان نامه کارشناسی ارشد، گروه مهندسی نرم افزار، دانشگاه فردوسی مشهد، ۱۳۸٤

[۹٥]. روح ا... مافی،برنامه نویسی اسکلتی موازی مبتنی بر عاملها با استفاده از معماری گرید، پایان نامه کارشناسی ارشد، گروه مهندسی نرم افزار، دانشگاه فردوسی مشهد، ۱۳۸۳

ضمائم

ضمیمه الف: فهرستی از پروژه های انجام شده و در حال انجام گرید

Category	Project	Organisation	Remarks
Integrated Grid Systems	NetSolve	U. Tennessee	A programming and runtime system for accessing high-performance libraries and resources transparently.
	Ninf	U. Tokyo	Functionality is similar to NetSolve.
	ST-ORM	UPC, Barcelona	A scheduler for distributed batch systems.
	SILVER	PPNL and U. of Utah	A scheduler for distributed batch systems.
	Albatross	Vrije U.	Object oriented programming system.
	PUNCH	Purdue U.	A portal computing environment and service for applications.
	Javelin	UCSB	Java-based programming and runtime system.
	XtremWeb	Paris-Sud U.	A global computing environment
	DISCWorld	U. of Adelaide	A distributed information-processing environment.
	Unicore	Germany	Java-based environment for accessing remote supercomputers.
	Alchemi	U. Of Melbourne	.NET based Desktop Grid System.
Core Middleware	Cosm	Mithral	A toolkit building P2P applications.
	Globus	ANL and ISI	Globus provides uniform and secure environment for accessing remote computational and storage resources.
	Gridbus	Melbourne University	Economic paradigm for service-oriented computing

	GridSim	Monash U.	A toolkit for Grid simulation.
	JXTA	Sun Microsystems	A Java-based framework and infrastructure for P2P computing.
	Legion	U. of Virginia	A Grid operating system providing transparent access to distributed resources.
	P2P Accelerator	Intel	A basic infrastructure for creating P2P applications for .NET platform.
User-level Middleware: Schedulers	AppLeS	UCSD	Application specific scheduler.
	Condor-G	U. of Wisconsin	A wide area job processing system.
	Nimrod-G	Monash U.	Economic-based Grid resource broker for parameter sweep/task farming applications.
User-level Middleware: Programming Environments	MPICH-G	Northern Illinois U.	MPI implementation on Globus.
	Nimrod parameter programming tools	Monash U.	A declarative language parametric programming.
	MetaMPICH	RWTH, Aachen	MPI programming and runtime environment.
	Cactus	Max Planck Institute for Gravitational Physics	A framework for writing parallel applications. It is developed using the MPICH-G and Globus.
	GrADS	Rice U.	Grid application development tools.
	GridPort	SDSC	Tools for creating computing portals.
Applications and application-driven Grid	European Data Grid	CERN	High Energy Physics, Earth Observation, Biology
	GriPhyN	UCF and ANL	High Energy Physics
	PPDG	Caltech and ANL	High Energy Physics

efforts	Virtual Laboratory	Monash U and WEHI	Molecular modeling for drug design.
	HEPGrid	Melbourne U	High Energy Physics apps.
	NEESGrid	NCSA	Earthquake Engineering
	Geodise	Southampton U.	Aerospace Design Optimisation
	Fusion Grid	Princeton/ANL/	Magnetic fusion
	IPG	NASA	Aerospace
	Active Sheet	Monash, QUT, & DSTC	Spread sheet processing
	Earth System Grid	LLNL, ANL, &NCAR	Climate Modeling
	Virtual Instruments	UCSD	Neuroscience
	National Virtual Observatory	Johns Hopkins U. & Caltech	Access to distributed astronomical databases and processing.

ضمیمه ب: فهرستی از سیستمهای مدیریت منابع موجود در گرید

[Tierney 2000] [Brooks 1997]	Intensive Distributed Computing Group, Lawrence Berkeley National Laboratory	server, which provides high-performance data handling and architecture for building high-performance storage systems from low-cost commodity hardware components. This technology has been quite successful in providing an economical, high-performance, widely distributed, and highly scalable architecture for caching large amounts of data that can potentially be used by many different users.	architecture: agents are processes that monitors the state of the system; broker agent (or broker) is an agent that manages the information, filters information for clients, or performs some action on behalf of a client. Agents model their environment using an extensible set of Facts and act on their environment using a set of Tasks. Features: object model; no QoS; agent-based store; centralised queries discovery; periodic push advertisement.
Globus [Foster1 997] [Czajko wski199 8]	Mathemati cs and Computer Science Division, Argonne National Laboratory	The Globus system is intended to achieve a vertically integrated treatment of application, middleware, and network. A low-level toolkit provides basic mechanisms such as communication, authentication, network information, and data access. These mechanisms are used to construct various higher-level metacomputing services, such as parallel programming tools and schedulers. The long-term goal is to build a grid infrastructure, an integrated set of higher-level services that enable applications to adapt to heterogeneous and dynamically changing metacomputing environments.	The architecture distributes the resource management problem among distinct local manager, resource broker, and resource co-allocator components, and defines an extensible resource specification language (RSL) to exchange information about requirements. The information service within the architecture uses a Metacomputing Directory Service (MDS) [Fitzgerald1997], which adopts the data representations and API defined by the LDAP service [Yeong1995]. Features: extensible schema model; soft QoS; network directory store; distributed queries discovery; periodic push advertisement.
GRACE [Buyya2 000]	School of Computer Science and Software Engineerin	GRACE (Grid Architecture for Computational Economy) is a new framework that uses economic theories in grid resource management and	Nimrod/G is a grid resource broker that allows managing and steering task farming applications (parameter studies) on computational grids. It follows an economic

	g, Monash University, Australia	scheduling. The components that make up GRACE include global scheduler (broker), bid-manager, directory server, and bid-server working closely with grid middleware and fabrics. The GRACE infrastructure also offers generic interfaces (APIs) that the grid tools and applications programmers can use to develop software supporting the computational economy.	(computational market) model for resource management and scheduling. It allows the study of the behaviour of output variables against a range of different input scenarios. Features: extensible schema model; hard QoS; relational resource info store; distributed queries discovery; periodic push/pull advertisement.
Legion [Grimshaw1999] [Chapin 1999]	Dept. of Computer Science, Univ. of Virginia	Legion is an object-oriented metacomputing environment, intended to connect many millions of hosts ranging from PCs to massively parallel supercomputers. It manages billions of objects and allows users to write and run applications in an easy-to-use, transparent fashion. It unites machines from thousands of administrative domains into a single coherent system.	Legion uses a resource management infrastructure. The philosophy of scheduling is that it is a negotiation of service between autonomous agents, one acting on the part of the application (consumer) and one on behalf of the resource or system (provider). The components of the model are the basic resources (hosts and vaults), the information database, the schedule implementer, and an execution monitor. Features: extensible object model; soft QoS; object model store; distributed queries discovery; periodic pull advertisement.
NetSolve [Casanova1998]	Dept. of Computer Science, Univ. of Tennessee	NetSolve is a client-server system that enables users to solve complex scientific problem remotely. The system allows users to access both hardware and software computational resources distributed across a network. NetSolve searches for computational resources on a network, chooses the best one available, and using retry	The NetSolve agent operates both as a database and as a resource broker. The agent keeps track of information about all the servers in its resource pool, including their availability, load, network accessibility, and the range of computational tasks that they can perform. The agent then selects a server to perform the task, and the server responds to the client's request.

Name	Unit	Project Description	Resource Management
Condor [Litzkow1988] [Raman 1998]	Condor team, Dept. of Computer Science, Univ. of Wisconsin-Madison	The goal of the Condor project is to develop, implement, deploy and evaluate mechanisms and policies that support High Throughput Computing (HTC) on large collections of distributively owned computing resources.	Condor uses a classified advertisement (classad) matchmaking framework for flexible resource management in distributed environments with decentralised ownership of resources, which uses the matchmaker/entity (which can be both provider and requestor) structure. Features: extensible schema model; no QoS; network directory store; centralised queries discovery; periodic push advertisement.
DPSS	Data	The DPSS is a data block	DPSS uses a broker/agent