www.Prozhe.com

دانشگاه جامع
علمی-کاربردی

وزارت علوم، تحقیقات و فناوری

مرکز آموزش علمی – کاربردی پارس آباد۱

تحت نظارت دانشگاه جامع – علمی کاربردی

پایان نامه دوره کاردانی ناپیوسته ترمی رشته نرم افزار کامپیوتر

عنوان پروژه :

درخت کشف منبع ردپا در محیط گرید

استاد راهنما :

عادل نادری

نگارش :

مهدی عزیزی

تاریخ تنظیم

نیمسال اول – پاییز ۹۱-۹۰

www.Prozhe.com

www.Prozhe.com

www.Prozhe.com

www.Prozhe.com

بسم الله الرحمن الرحيم

www.Prozhe.com

www.Prozhe.com

وزارت علوم، تحقیقات و فناوری

مرکز آموزش علمی – کاربردی پارس آباد۱

تحت نظارت دانشگاه جامع – علمی کاربردی

پایان نامه دوره کاردانی ناپیوسته ترمی رشته نرم افزار کامپیوتر

عنوان پروژه :

درخت کشف منبع ردوپا در محیط گرید

استاد راهنما :

عادل نادری

نگارش :

مهدی عزیزی

تاریخ تنظیم

نیمسال اول – پاییز ۹۱-۹۰

www.Prozhe.com

www.Prozhe.com

Ministry of Science, Research and Technology of Islamic Republic of IRAN
University of Applied Science and Technology
Parsabad۱ College

Course
Associate degrees of Computer Software

Project Title
Tree in the track grid resource discovery

Supervisor
A.naderi

Authors
Mehdi azizi

First Semester - Fall ۹۰-۹۱

www.Prozhe.com

www.Prozhe.com

ارزشیابی توسط اساتید

این گزارش در تاریخ ..... مورد ارزیابی قرار گرفته و کار انجام شده از نظر مضمون و کیفیت به عنوان دو/سه واحد پروژه کاردانی(ترمی) نرم افزار پذیرفته شده است.

ردیف	عنوان	نام و نام خانوادگی	امضاء
۱	استاد پروژه	عادل نادری	
۲	مسئول کمیته تخصیص پروژه		
۳	استاد داور		
۴	عضو کمیته تخصیص پروژه		

www.Prozhe.com

www.Prozhe.com

سپاسگزاری

با سپاس و قدردانی فراوان از استاد بزرگوارم عادل نادری ، ریاست محترم و مسئول

آموزش که همواره از راهنمایی‌های ایشان بهره‌مند شدم

www.Prozhe.com

www.Prozhe.com

تقدیم به

پدر بزرگوار و مادر مهربانم
آن دو فرشـــته ای کـــه از خواســـته هایشان گذشتند ، ســختی هــا را بــه جان خریدنــد و خــود را ســپر بــلای مشکلات و ناملایمات کردند تا من بــه جایگاهی که اکنون در آن ایــستاده ام برسم.

www.Prozhe.com

www.Prozhe.com

فهرست مطالب

عنوان | صفحه

چکیده فارسی ۱

مقدمه ۲

فصل اول

مقدمه ای بر گرید و بحث کشف منبع در آن ۴

۱-۱- مقدمه ۴

۱-۲- مروری بر گرید ۴

۱-۳- آشنایی با مسائل و چالش‌های مهم موجود در گرید ۶

۱-۴- مفهوم کشف منبع در گرید ۶

۱-۵- روش‌های استفاده شده برای کشف منبع در گرید ۸

۱-۶- دید کلی از روش پیشنهادی خودمان ۱۰

۱-۷- خلاصه و نتیجه‌گیری ۱۰

فصل دوم

بررسی کارهای مربوطه در زمینه کشف منبع ۱۳

۲-۱- مقدمه ۱۳

۲-۲- کشف منبع در گرید با استفاده از Re-routing Tables ۱۳

۲-۳- روش پیشنهادی Li **برای کشف منبع در محیط گرید** ۱۶

۲-۴- کشف منبع با استفاده از مقادیر اعتبار برای سازمان‌های مجازی ۱۸

۲-۵- روش کشف منبع با استفاده از نقشه بیتی در گرید ۲۰

۲-۶- سایر روش‌های کشف منبع ۲۲

۲-۷- تفاوت روش پیشنهادی خودمان با روش‌های بحث شده ۲۳

۲-۸- نتیجه گیری ۲۴

فصل سوم

ارائه الگوریتم پیشنهادی ۲۷

۳-۱- مقدمه ۲۷

www.Prozhe.com

www.Prozhe.com

۳-۲- توصیف نقشه‌های بیتی استفاده شده در الگوریتم ۲۷
۳-۳- تشکیل درخت و توصیف الگوریتم ۲۹
۳-۴- الگوریتم کشف منبع پیشنهادی (FRDT) ۳۴
۳-۵- الگوریتم به روز رسانی ۳۶
۳-۶- بهبود روش در یک محیط گرید واقعی ۳۹
۳-۷- پیچیدگی مکانی و زمانی ۴۱
۳-۸- خلاصه و نتیجه‌گیری ۴۱
فصل چهارم
نتایج شبیه‌سازی ۴۴
۴-۱- مقدمه ۴۴
۴-۲- نتایج شبیه سازی ۴۵
۴-۳- نتیجه‌گیری ۵۰
فصل پنجم
vhdl ۵۳
۵-۱- مقدمه ای بر vhdl ۵۳
۵-۲- فاکتورهای قابل ملاحظه در ارزش vhdl ۵۳
۵-۳- تعریف ورودی و خروجی ۵۵
۵-۴- تحلیل یک برنامه vhdl ۵۷
۵-۵- انواع داده ها data types ۵۸
۵-۶- تعریف فیزیکی یک نوع داده ۵۹
فصل ششم
نتیجه‌گیری و کارهای آینده ۶۲
۶-۱- نتیجه‌گیری ۶۲
۶-۲- کارهای آینده ۶۲
فهرست منابع (ابتدا منابع فارسی و سپس منابع غیر فارسی) ۶۳
منابع فارسی ۶۳

www.Prozhe.com

www.Prozhe.com

منابع لاتین ۶۴
چکیده انگلیسی ۶۶

www.Prozhe.com

www.Prozhe.com

فهرست جداول

جدول | صفحه

شکل ۲-۱: نمونه‌ای از محیط گرید تشکیل شده از مسیریاب‌ها و منابع ۱۴
شکل ۲-۲: نمونه‌ای از یک جدول مسیریابی ۱۵
شکل ۲-۳: مثالی از یک کشف منبع ۱۶
شکل ۲-۴: نمونه‌ای از توپولوژی شبکه..... ۱۷
شکل ۲-۵: مثالی از کشف منبع ۱۸
شکل ۲-۶: نمونه‌ای از یک محیط گرید فرضی متشکل از سازمان‌های مجازی..... ۱۹
شکل ۲-۷: مثالی از کشف منبع بین گره‌های محیط ۲۱
شکل۳ -۱: نمونه‌ای از یک نقشه بیتی مربوط به سیستم عامل ۲۸
شکل ۳-۲: نمونه‌ای از درخت کشف منبع ردپا قبل از مرحله آماده سازی..... ۳۰
شکل ۳-۳: جمع آوری اطلاعات فرزندان توسط گره ۶ (مربوط به MacOS) ۳۱
شکل ۳-۴: جمع آوری اطلاعات فرزندان توسط گره ۱ (مربوط به MacOS) ۳۲
شکل ۳-۵: الگوریتم تشکیل نقشه‌های بیتی مسیر و شمارنده ۳۳
شکل ۳-۶: نمونه‌ای از درخت کشف منبع ردپا بعد از مرحله آماده‌سازی..... ۳۴
شکل ۳-۷: نمونه‌ای از یک فرم درخواست..... ۳۴
شکل ۳-۸: نمونه‌ای از فرآیند کشف منبع در درخت کشف منبع وزن‌دار..... ۳۶
شکل ۳-۹: نمونه‌ای از فرآیند به روز رسانی..... ۳۸
شکل ۳-۱۰: نمونه‌ای از یک درخت کشف منبع ردپا با مقدار آستانه ۲ ..... ۴۰
شکل ۴-۱: تعداد گره‌هایی که توسط درخواست‌های کشف منبع ملاقات می‌شوند ۴۵
شکل ۴-۲: تعداد گره‌هایی درخواست‌های کاربران به آن‌ها فرستاده می‌شوند ۴۶
شکل ۴-۳: مجموع تعداد گره‌های ملاقات شده در مراحل کشف منبع و به روز رسانی ۴۷
شکل ۴-۴: متوسط تعداد گره‌هایی که توسط درخواست‌های کشف منبع برای روش‌های مختلف ملاقات می‌شوند..... ۴۸

www.Prozhe.com

www.Prozhe.com

شکل ۴-۵: تعداد لینک‌های ملاقات شده توسط درخواست‌های کشف منبع و به روز رسانی به ازای ۳۰۰ درخواست در محیط ۴۹
شکل ۴-۶: تعداد لینک‌های ملاقات شده توسط درخواست‌های کشف منبع و به روز رسانی به ازای ۱۰۰۰ درخواست در محیط ۴۹
شکل ۴-۷: تعداد لینک‌های ملاقات شده توسط درخواست‌های کشف منبع به ازای ۳۰۰ درخواست توسط روش‌های مختلف ۵۰

www.Prozhe.com

www.Prozhe.com

چکیده

در این پایاننامه ما یک الگوریتم جدید پیشنهاد می‌کنیم و آن استفاده از یک درخت وزن‌دار برای کشف منبع است. ما از یک نقشه‌بیتی که تعداد خانه‌های آن به نسبت تعداد صفت‌های موجود برای هر منبع است استفاده می‌کنیم. در واقع ما در این درخت در هر گره ردپایی از منابع موجود در فرزندان یا نسل‌های آن گره را خواهیم داشت و در نتیجه با رسیدن درخواست کاربر به هر یک از این گره‌ها در صورت وجود این منبع درخواست شده در فرزندان یا نسل‌ها، ما با استفاده از این ردپا می‌توانیم به صورت مستقیم و بدون مراجعه به گره‌های اضافی و ایجاد ترافیک غیرضروری به گره‌های که این منبع را در اختیار دارد دست پیدا کنیم.

واژگان کلیدی : درخت وزن دار ، کشف منبع ، نقشه بیتی ، ردپایی در هر گره از فرزندان ، نسل های گره.

www.Prozhe.com

www.Prozhe.com

مقدمه

مقدمه‌ای در مورد پیدایش سیستم‌های گرید (رایانش مشبک) و یکسری از بحث‌های کلی در مورد مسئله کشف منبع در محیط گرید بیان شده است. در ادامه مروری بر گرید ، آشنایی با مسائل و چالش‌های مهم موجود در گرید، مفهوم اکتشاف منابع در این محیط، روش‌هایی که برای کشف منبع در محیط گرید استفاده می‌شوند، دید کلی از روش پیشنهادی خودمان و یک ارزیابی کلی از روش پیشنهادی در این پایاننامه آورده شده است. این فصل با خلاصه و نتیجه‌گیری از فصل و نیز تشریح ساختار پایاننامه به پایان رسیده است.

هنگامی که کاربر در محیط گرید منبعی را درخواست می کند، یک الگوریتم کشف منبع باید منبع درخواستی را پیدا کرده و در اختیار کاربر قرار دهد. روش‌های مختلفی برای کشف منبع در محیط گرید پیشنهاد شده‌اند. این راه حل‌ها را می‌توان به دو گروه متمرکز و نامتمرکز تقسیم کرد. روش‌های متمرکز روش‌هایی هستند که وابسته به یک سرور مرکزی هستند در حالی که روش‌های نا متمرکز اینگونه نیستند.

الگوریتم‌های کشف منبعی که برای محیطی مانند گرید، که تعداد منابع و درخواست‌ها بسیار زیاد است، پیشنهاد می‌شوند باید فاکتورهایی مانند هزینه بسیار پایین بر کشف منبع و به روز رسانی، ترافیک تولیدی بسیار کم در مراحل آماده‌سازی، کشف و به روز رسانی، مقیاس‌پذیری، سادگی روش بدون نیاز به محاسبات پیچیده، فضای مورد نیاز کم برای ذخیره اطلاعات (هر چقدر اطلاعات کمتر و منظم‌تر باشند مدیریت آنها آسان‌تر است)، را در نظر بگیرند.

در شبیه‌سازی‌ها ما همانند مراجع (Mastroianni, ۲۰۰۷) و (Chang, ۲۰۱۰) ارتفاع درخت‌ها را ۳ و ۴ در نظر گرفتیم و شبیه‌سازی‌ها را بر روی درخت‌هایی با ارتفاع ۳ و ۴ انجام دادیم. در آزمایشات ما تعداد گره‌های موجود در محیط را متغیر در نظر گرفته و همچنین تعداد درخواست‌های مشخص را به صورت تصادفی به گره‌های موجود در محیط تحویل می‌دهیم. ما در شبیه‌سازی‌ها تعداد گره‌های ملاقات شده در روش‌های مختلف را به ازای درخواست‌های مشخص محاسبه کرده و در نمودارها نشان داده‌ایم. همانطوری که مشخص است هر چقدر در یک روش تعداد گره‌های ملاقات شده در مراحل کشف منبع یا به روزرسانی کمتر باشد آن روش موثرتر است.

www.Prozhe.com

www.Prozhe.com

فصل اول

www.Prozhe.com

www.Prozhe.com

مقدمه‌ای بر گرید و بحث کشف منبع در آن

۱-۱- مقدمه

در این فصل، مقدمه‌ای در مورد پیدایش سیستم‌های گرید[۱] (رایانش مشبک) و یکسری از بحث‌های کلی در مورد مسئله کشف منبع در محیط گرید بیان شده است. در ادامه مروری بر گرید ، آشنایی با مسائل و چالش‌های مهم موجود در گرید، مفهوم اکتشاف منابع در این محیط، روش‌هایی که برای کشف منبع در محیط گرید استفاده می‌شوند، دید کلی از روش پیشنهادی خودمان و یک ارزیابی کلی از روش پیشنهادی در این پایاننامه آورده شده است. این فصل با خلاصه و نتیجه‌گیری از فصل و نیز تشریح ساختار پایاننامه به پایان رسیده است.

۱-۲- مروری بر گرید

در اواسط دهه ۱۹۸۰، دو پیشرفت درفن آوری باعث تغییر وضعیت در روند طراحی سیستم‌های کامپیوتری شدند:

- اول تولید مایکرو کامپیوترهای قدرتمند بود که بسیاری از آن‌ها قدرت یک Mainframe را داشتند در حالی که قیمت آن‌ها بسیار کمتر بود. در واقع از کامپیوترهایی در دهه ۱۹۶۰، که حدوداً ۱۰۰ میلیون دلار قیمت داشتند و تنها تعداد کمی دستورالعمل در هر ثانیه اجرا می‌کردند به ماشین‌هایی رسیدیم که حدوداً ۱۰۰۰ دلار قیمت دارند و قادرند ۱۰ میلیون دستورالعمل در ثانیه اجرا کنند.
- دوم اختراع شبکه‌های کامپیوتری پر سرعت بود. شبکه‌های کامپیوتری گسترده به ما این امکان را دادند که بتوانیم میلیون‌ها ماشین در سرتاسر جهان را با سرعت بالا به یکدیگر متصل کنیم.

نتیجه این دو فن آوری (مایکرو کامپیوترها و فن آوری شبکه‌های کامپیوتری) این شد که بتوان با اتصال تعداد بسیار زیادی از کامپیوترها در یک شبکه پر سرعت به یکدیگر، سیستم های محاسباتی با کارایی بسیاربالا (حتی بسیار کاراتر از بهترین ابرکامپیوتر[۲] های سنتی) به دست آورد و در عین حالی که استقلال

---

[۱] Grid
[۲] Supercomputer

www.Prozhe.com

www.Prozhe.com

هر کامپیوتر حفظ می‌شود و هر کامپیوتر، پردازش‌های مالکش را اجرا می‌کند، از اتصال این کامپیوترهای مستقل، بتوان به توان محاسباتی بالا دست یافت. این کار از طریق یک لایه اضافی از نرم افزار، به نام میان افزار[۳] که بر روی سیستم عامل‌های کامپیوترهای مستقل نصب می‌شود، انجام می‌گیرد. معمولاً کل این ساختار، یک سیستم توزیع شده نامیده می‌شود (فتحی، ۱۳۸۸)

محاسبات گرید یک فن آوری جدید است که با استفاده از زیرساخت‌های ارتباطی و شبکه‌های کامپیوتری و نیز با بکارگیری حد اعلای مفاهیم و امکانات سیستم‌های توزیع شده، به ما امکان دسترسی به انواع منابع را به صورت راه دور می‌دهد. امروزه می‌توان با استفاده از فن آوری گرید، منابع محاسباتی ناهمگون را به یکدیگر وصل کرد به طوری که این ساختار همانند یک ماشین مجازی یکپارچه و تکی به نظر بیاید و سپس برنامه‌های کاربردی بسیار پیچیده و بزرگ را که به توان پردازشی بسیار بالا و حجم عظیمی از داده‌های ورودی نیاز دارند، بر روی این ماشین مجازی اجرا کرد. به طور مثال با استفاده از این فن آوری می- توان چندین ابر کامپیوتر، خوشه[۴] و کامپیوتر خانگی[۵]، همگی را به یکدیگر وصل کرد. گرید به لحاظ محدوده جغرافیایی و نوع منابع تحت پوشش هیچ محدودیتی ندارد. ممکن است یک سنسور پنج میلیمتری در سنگاپور و یک ابر کامپیوترغول پیکر در تگزاس آمریکا و یک کامپیوتر خانگی در ملبورن استرالیا همگی تحت پوشش یک گرید واحد باشند.

در حالت کلی‌تر، یک شبکه گرید می‌تواند چندین خوشه بزرگ، چندین ریز پردازنده[۶] از انواع مختلف و هزاران کامپیوتر شخصی و ایستگاه کاری[۷] در سراسر دنیا را در بر بگیرد، امکان اتصال و ارتباط سطح بالای این کامپیوترهای نامتجانس که با یکدیگر همخوانی ندارند را برقرار کند، منابع از انواع مختلف را شناسایی و دستیابی راه دور به آن‌ها را مدیریت کند و در نهایت یک بستر قوی جهت اجرای پردازش‌های بسیار عظیم و پیچیده را که به حجم عظیمی از منابع پردازشگر، دستگاه‌های ذخیره سازی اطلاعات در مقیاس وسیع، داده- های معمولاً توزیع شده و حتی دستگاه‌های خاص علمی نیاز دارند ارائه کند. در واقع با اتصال به یکی از

---

[۳] Middleware
[۴] Cluster
[۵] Personal Computer
[۶] Multiprocessor
[۷] Work station

www.Prozhe.com

www.Prozhe.com

شبکه‌های گرید می توان به توان محاسباتی بسیار بالا جهت اجرای پروژه‌های بسیار عظیم دست یافت. از طرف دیگر، گرید، تشریک مساعی و همکاری سازمان‌های مختلف در سرتاسر دنیا را در زمینه پروژه‌های مشترک امکان پذیر می‌سازد و بستری بسیار مناسب جهت به تحقق پیوستن سازمان‌های مجازی پویا ایجاد می نماید. در بخش بعد با مسائل و چالش‌هایی که هم اکنون در گرید، با آن‌ها مواجه هستیم خلاصه وار آشنا می‌شویم و بحث اصلی این پایاننامه یعنی مسئله کشف منبع را نیز در بخش‌های بعدی معرفی می‌کنیم.

۱-۳- آشنایی با مسائل و چالش های مهم موجود در گرید

مسائل و چالش‌های مختلفی درمحیط گرید وجود دارند که محققین همواره در تلاش بوده‌اند راه حل‌هایی را در جهت رفع و بهبود این مسائل ارائه دهند. از جمله مسائل و چالش‌هایی که در محیط گرید وجود دارند می‌توان به مسئله امنیت، مسئله اقتصادی، مدیریت سیستم، مدیریت شبکه، دستیابی واحد و یکسان، زمانبندی، اکتشاف منابع و غیره اشاره کرد. موضوعی که در این پایاننامه بر روی آن کار شده است، بحث اکتشاف منابع در محیط گرید است که یکی از موضوعات مهم و کلیدی در این محیط محسوب می شود. در بخش بعد مفهوم کشف منبع در محیط گرید توضیح داده می‌شود.

۱-۴- مفهوم کشف منبع در گرید

همانطوری که در بخش‌های قبلی نیز ذکر شد، مسائلی که توسط گرید حل می‌شوند معمولاً مسائلی هستند که نیاز به حجم محاسباتی بالا و یا حجم زیاد داده و ذخیره‌سازی دارند و ممکن است داده‌های ورودی مورد نیاز آن‌ها، در میان سایت‌های مختلف توزیع شده باشد. جهت حل این مسائل، گرید یک بستر محاسباتی/ ارتباطی را ارائه می‌کند که در آن، قابلیت‌ها و امکانات و کیفیت سرویس کلیه منابع و بالاخص منابع محاسباتی در کنار یکدیگر مجتمع شده است و دیدی از یک ماشین مجازی بسیار قدرتمند در اختیار برنامه-های کاربردی جهت اجرا قرار می‌گیرد. بنابراین، گرید هیچ محدودیتی از نظر محدوده جغرافیایی تحت پوشش ندارد و منابع در دسترس یک گرید، به لحاظ جغرافیایی توزیع شده هستند و ممکن است ناهمگن نیز باشند، منابع در سایت‌های مختلف قرار داشته باشد و متعلق به سازمان‌ها و نهادهای متفاوت باشند. نوع

www.Prozhe.com

www.Prozhe.com

منابعی که در محیط گرید به اشتراک گذاشته می‌شوند ممکن است از نوع سیستم‌های رومیزی[۸]، خوشه‌ها، مجموعه داده‌های بزرگ، قطعات ذخیره سازی و غیره باشند. سوالی که در اینجا مطرح است این است که هنگامی که یک کاربر راه دور به یک منبع راه دور برای اجرای یک کار یا دسترسی به داده‌های آن منبع نیاز داشته باشد، چگونه به این منبع دسترسی پیدا می‌کند؟ در واقع مکانیزمی که بوسیله ساختار گرید فراهم می‌شود باید قادر باشد منبع مناسب را برای کاربر کشف کند. در نتیجه یکی از اساسی‌ترین کارها در یک گرید، پشتیبانی کردن از مکانیزم کشف منبع است(Buyya, ۲۰۰۵; Karaoglanoglou, ۲۰۰۸).

هدف اکتشاف منابع، شناسایی لیستی از منابع در دسترس جهت واگذاری کارها می‌باشد. وقتی یک کاربر در محیط گرید برای اجرای یک برنامه خاص، منبعی را درخواست می‌کند، یک روش کشف منبع باید بتواند منبع مورد درخواست کاربر را با هزینه زمانی و مکانی کم برای کاربر کشف کند تا آن کاربر بتواند هر چه سریعتر از منبع استفاده کند. اگر الگوریتم کشف منبعی بتواند با ایجاد کمترین ترافیک در محیط، بدون مراجعه به گره‌های غیر ضروری و اضافی و در یک زمان کوتاه، منبع مطابق با درخواست کاربر را کشف کند، خواهد توانست رضایت کاربران را جلب کند و کارایی سیستم به طور قابل ملاحظه‌ای افزایش خواهد یافت. در نتیجه چنین الگوریتمی می تواند یک الگوریتم موثر و قابل استفاده در یک محیط واقعی گرید باشد.

کشف یک منبع خاص در محیط های محاسباتی سنتی کار چندان مشکلی نبود چرا که تعداد منابع به اشتراک گذاشته شده کم بوده و تمام منابع تحت کنترل یک مرکز بودند. در سیستم‌های گرید، چندین فاکتور اصلی وجود دارند که مسئله کشف منبع را تبدیل به یک مسئله بسیار مشکل می کنند. این فاکتورها عبارتند از: تعداد منابع بسیار زیاد، مالکیت توزیع شده، ناهمگن بودن منابع، خرابی منبع و غیره. به عبارت دیگر هر یک از روش‌هایی که برای کشف منبع در محیط گرید پیشنهاد می‌شوند با این مشکلات مواجه هستند. یکی دیگر از مسائل بسیار مهم در گرید که باید در مکانیزم‌های کشف منبع در نظر گرفته شود، پویایی گرید است. منابع به اشتراک گذاشته شده در گرید، ممکن است در دو حالت ممکن باشند: در دسترس[۹] و خارج از دسترس[۱۰].

---

[۸] Desktop
[۹] Online
[۱۰] Offline

www.Prozhe.com

www.Prozhe.com

تمام منابع موجود در گرید همواره در دسترس نیستند. یک منبع ممکن است در طول زمان از حالت در دسترس بودن به حالت خارج از دسترس برود و یا برعکس. مهمترین دلیل این امر نیز مالکیت توزیع شده در گرید است، یعنی ممکن است یک فراهم کننده منبع، منبع اش را در طول یک دوره زمانی معین در اختیار یک کاربر قرار دهد و بعد از آن منبع اش را از کاربر پس بگیرد. از علل دیگر تغییر وضعیت منابع می‌توان به خرابی منبع اشاره کرد که منبع را از حالت در دسترس بودن به حالت خارج از دسترس می‌برد. بر عکس این حالت نیز ممکن است. یعنی همواره ممکن است منابع به یک سیستم گرید اضافه و کم شوند. در تمامی این حالات باید اطلاعات این منابع به درستی و بلافاصله دراختیار باشد تا مکانیزم کشف منبع بتواند بر اساس این اطلاعات به روز شده، منابع صحیحی را برای کاربران کشف کند. واضح است که این کار در محیط وسیع و پویایی مانند گرید کار بسیار مهم و در عین حال یک کار پیچیده و مشکل است.

حال که با مفهوم کشف منبع و مسائل موجود در کشف منبع به طور خلاصه وار آشنا شدیم، در بخش بعد با روش‌هایی که اکثر الگوریتم‌های کشف منبع از آن روش‌ها در کارهای خود استفاده می‌کنند، آشنا می‌شویم.

۱-۵- روش‌های استفاده شده برای کشف منبع در گرید

الگوریتم‌های کشف منبع در محیط گرید از روش‌های متفاوتی برای این کار استفاده می‌کنند. بسیاری از الگوریتم‌ها برای کشف منبع درمحیط گرید از روش متمرکز[۱۱] به دلیل سادگی این روش استفاده می‌کنند (Foster, ۱۹۹۷; Mutka, ۱۹۸۷; Germain, ۲۰۰۰; Chien, ۲۰۰۳; Berman, ۲۰۰۳; Neary, ۱۹۹۹).

الگوریتم‌هایی که از روش متمرکز در کارهایشان استفاده می‌کنند دارای مزایایی همچون مدیریت آسان گره-ها هستند. دلیل این امر آن است که یک سرور مرکزی تمام اطلاعات منابع و گره‌ها را دراختیار دارد و تمامی گره‌ها تحت نظارت یک سرور مرکزی هستند. واضح است که این روش برای محیط هایی مناسب‌تر است که تعداد گره‌های موجود در آن محیط، تعداد درخواست‌های رسیده به سرور و تنوع منابع در آن محیط کم باشد. با افزایش تعداد گره‌های محیط و افزایش تعداد درخواست‌ها در آن محیط، تمامی درخواست‌ها به سرور مرکزی فرستاده می‌شوند، درناحیه سرور گلوگاه[۱۲] ایجاد می‌شود و کارایی سیستم به شدت پایین می‌آید. از

[۱۱] Centralized approach
[۱۲] bottleneck

www.Prozhe.com

www.Prozhe.com

دیگر ایرادات این روش‌ها نیز می‌توان به نقاط تنهای خراب[۱۳] اشاره کرد، بنابراین روش‌های متمرکز برای محیط‌های وسیعی همچون گرید مناسب نیستند.

از روش‌های دیگری که الگوریتم‌های کشف منبع از آن‌ها استفاده می‌کنند و توانسته بسیاری از ایرادات روش‌های متمرکز را برطرف کند روش‌های نامتمرکز[۱۴] است. این روش‌ها بر خلاف روش‌های متمرکز دارای یک سرور مرکزی نیستند و بنابراین مشکلات روش‌های متمرکز را نخواهند داشت. از جمله روش‌هایی که از الگوریتم‌های توزیع شده درکارهای خود استفاده کرده‌اند می‌توان به روش سیل‌آسا [۱۵] و روش تصادفی [۱۶] اشاره کرد. روش سیل‌آسا در اکثر الگوریتم‌های کشف منبع گرید، به دلیل سادگی این روش، استفاده می‌شود. در این روش درخواست‌ها به صورت سیل‌آسا به تمامی گره‌های محیط فرستاده می‌شوند. این روش کشف یک منبع خاص را در محیط به شرطی که آن منبع در محیط وجود داشته باشد تضمین می‌کند، ولی ایراد بزرگ این روش که باعث شده است این روش یک روش مناسب برای محیطی همچون گرید نباشد، ترافیک سنگین آن است.

روش دیگری که نسبت به روش سیل‌آسا ترافیک کمتری را در محیط ایجاد می‌کند روش تصادفی است. در این روش نیز درخواست‌ها به مسیرهای تصادفی فرستاده می‌شوند ولی این روش نیز دارای تاخیر زیاد می‌باشد که برای محیطی وسیع مانند گرید مناسب نیست.

از الگوریتم‌های دیگری که برای کشف منبع از روش نامتمرکز استفاده می‌کنند می توان به روش- های مبتنی بر درخت اشاره کرد. این روش‌ها که اخیراً ارائه شده‌اند برای کشف منبع از ساختار درختی استفاده می‌کنند، یعنی گره‌های موجود در محیط گرید را به صورت درختی به یکدیگر متصل می‌کنند و کشف منبع را بر روی این ساختار انجام می‌دهند. این روش‌ها توانسته‌اند بسیاری از ایرادات روش‌های قبلی از جمله ترافیک سنگین را برطرف کنند. از جمله این روش‌ها می‌توان به روشی با نام " A resource discovery tree using bitmap for grids" (Chang, ۲۰۱۰) اشاره کرد که اخیراً ارائه شده است. این روش دارای هزینه کشف

---

[۱۳] Single points of failure
[۱۴] Distributed approaches
[۱۵] Flooding-Based
[۱۶] Random-Based

www.Prozhe.com

www.Prozhe.com

منبع و به روز رسانی[۱۷] بسیار کمتری نسبت به روش‌های قبلی می‌باشد ولی این روش نیز دارای ایراداتی می‌باشد که در فصل بعد به طور مفصل مورد بررسی قرار می‌گیرد.

۱-۶- دید کلی از روش پیشنهادی خودمان

همانگونه که دربخش‌های قبلی نیز بیان شد برخی از روش‌های کشف منبع که اخیراً ارائه شده‌اند، از روش درختی برای این کار استفاده می‌کنند. الگوریتم کشف منبعی که ما در این پایاننامه پیشنهاد داده‌ایم نیز از یک ساختار درختی استفاده می‌کند با این تفاوت که ما برای اولین بار از یک ساختار درخت وزن‌دار [۱۸] برای کشف منبع در محیط گرید استفاده کرده‌ایم. ما با استفاده از وزن‌های درخت و روشی که برای کشف منبع پیشنهاد داده‌ایم، توانسته‌ایم روش‌های قبلی را تا حد بسیار زیادی بهبود دهیم. در این کار ما از یک نقشه بیتی [۱۹] برای مشخص کردن منابع موجود در گره‌ها و همچنین منابع موجود در فرزندان و نسل‌های یک گره استفاده می‌کنیم. ما در این درخت ردپایی [۲۰] از منابع را در گره‌های محیط ذخیره می‌کنیم تا در صورت رسیدن هر درخواست به هر یک از این گره‌ها، مکانیزم بتواند با استفاده از این اعداد به طور مستقیم به گره‌ای که حاوی منبع مورد نظر کاربر است بدون سرکشی به گره‌های اضافی و غیر ضروری و اتلاف زمان دسترسی پیدا کند. برای ارزیابی کارایی الگوریتم، ما روش خودمان را به لحاظ تعداد گره‌های ملاقات شده و ترافیک تولیدی با چند روش دیگر مقایسه کرده‌ایم. برای این کار از یک سری شبیه‌سازی‌ها استفاده شده است و نتایج در قالب نمودارهایی آورده شده است. نتایج نشان می‌دهند که روش پیشنهادی ما نسبت به روش‌های دیگر بسیار موثرتر است و می‌تواند تا حد زیادی از ترافیک تولیدی در سیستم بکاهد.

۱-۷- خلاصه و نتیجه‌گیری

اساس تکنولوژی گرید مفهوم اشتراک منابع است. در واقع محیط گرید یک محیط محاسباتی توزیع‌شده مجازی است که هدف اصلی آن به اشتراک گذاشتن منابعی است که به لحاظ جغرافیایی توزیع شده هستند.

---

[۱۷] Update
[۱۸] Weighted tree structure
[۱۹] Bitmap
[۲۰] Footprint

www.Prozhe.com

www.Prozhe.com

برای اینکه در چنین محیطی یک کاربر راه دور بتواند از منابع راه دور موجود در محیط استفاده کند، ابتدا باید منابع مورد نیاز کاربر را که در محیط در دسترس هستند برای کاربر کشف شوند. الگوریتم‌های کشف منبع برای چنین هدفی به وجود آمده‌اند. این الگوریتم‌ها وظیفه کشف منابع درخواستی کاربران را از محیط بر عهده دارند. واضح است که کشف منبع در محیط گسترده و پویایی همچون گرید که ممکن است حتی میلیون‌ها منبع در محیط در فواصل جغرافیایی دور از یکدیگر پراکنده شده باشند، کار بسیار پیچیده و دشواری است که محققین همواره در پی یافتن راه‌حلی بهینه برای این مسئله بوده‌اند. الگوریتم‌هایی برای کشف منبع در این محیط می‌توانند قابل استفاده و کارا باشند که بتوانند با ایجاد کمترین ترافیک، منبع مورد درخواست کاربر را بدون هدر دادن زمان برای کاربر کشف کنند. همچنین این الگوریتم‌ها باید قادر باشند تغییرات وسیعی را که در منابع این محیط به صورت مکرر اتفاق می‌افتد را به سرعت تشخیص داده و با هزینه زمانی و مکانی کم و با ایجاد ترافیک پایین، محیط را به روز رسانی کنند. روشی را نیز که ما در این پایاننامه ارائه داده‌ایم روش کشف منبعی است که دارای هزینه کشف منبع و به روز رسانی بسیار کمتری نسبت به روش‌های قبلی است. ما با استفاده از این روش می‌توانیم منابع درخواستی کاربر را با کمترین تعداد گره‌های ملاقات شده و با ایجاد کمترین ترافیک در محیط برای کاربر کشف کنیم. نتایج شبیه سازی‌ها نیز این مطلب را به وضوح نشان می‌دهند.

در ادامه ساختار پایاننامه آورده شده است. در فصل دوم برخی از کارهایی را که در زمینه کشف منبع در محیط گرید اخیراً ارائه شده‌اند را بحث و بررسی کرده‌ایم. همچنین تفاوت روش خودمان با این روش‌ها نیز در انتهای فصل آورده شده است. در فصل سوم روش پیشنهادی کشف منبع خودمان را به تفصیل بحث و بررسی می‌کنیم. در فصل چهارم نتایج شبیه‌سازی‌ها برای کار خودمان آورده شده است. همچنین در این فصل روش خودمان را با روش‌های دیگر با ارائه نمودارهایی مقایسه خواهیم کرد. در نهایت فصل پنجم شامل نتیجه‌گیری و کارهای آینده است.

www.Prozhe.com

www.Prozhe.com

فصل دوم

www.Prozhe.com

www.Prozhe.com

بررسی کارهای مربوطه در زمینه کشف منبع

۲-۱- مقدمه

هنگامی که کاربر در محیط گرید منبعی را درخواست می کند، یک الگوریتم کشف منبع باید منبع درخواستی را پیدا کرده و در اختیار کاربر قرار دهد. روش‌های مختلفی برای کشف منبع در محیط گرید پیشنهاد شده‌اند. این راه حل‌ها را می‌توان به دو گروه متمرکز و نامتمرکز تقسیم کرد. روش‌های متمرکز روش‌هایی هستند که وابسته به یک سرور مرکزی هستند در حالی که روش‌های نا متمرکز اینگونه نیستند. همانطوری که قبلا نیز بحث شد در روش‌های متمرکز در محیطی مانند گرید که تعداد درخواست‌های کاربران و تعداد گره‌های موجود در گرید بسیار زیاد است، در ناحیه سرور گلوگاه ایجاد شده و کارایی کل سیستم کاهش می‌یابد. روش‌های نامتمرکزی هم که تاکنون ارائه شده‌اند نتوانسته‌اند روش‌های کشف منبعی را ارائه دهند که منبع مورد نیاز کاربر را بدون مراجعه به گره‌های اضافی و غیر ضروری کشف کنند. در ادامه برخی از روش‌های کشف منبعی که اخیرا ارائه شده‌اند را هر کدام را در یک بخش توضیح خواهیم داد.

۲-۲- کشف منبع در گرید با استفاده از Re-routing Tables

این روش توسط karaoglanoglu و همکاران (۲۰۰۸) ارائه شده است. آنها یک روش کشف منبع در محیط گرید پیشنهاد داده‌اند که از یکسری جدول مسیریابی[۲۱] برای این کار استفاده می‌کند. در این روش محیط گرید به صورت ترکیبی از مسیریاب[۲۲]ها و منابع در نظر گرفته شده است و کشف منبع بین این مسیریاب ها انجام می گیرد. نمونه‌ای از این محیط را می‌توانید در شکل ۲-۱ ملاحظه می‌کنید.

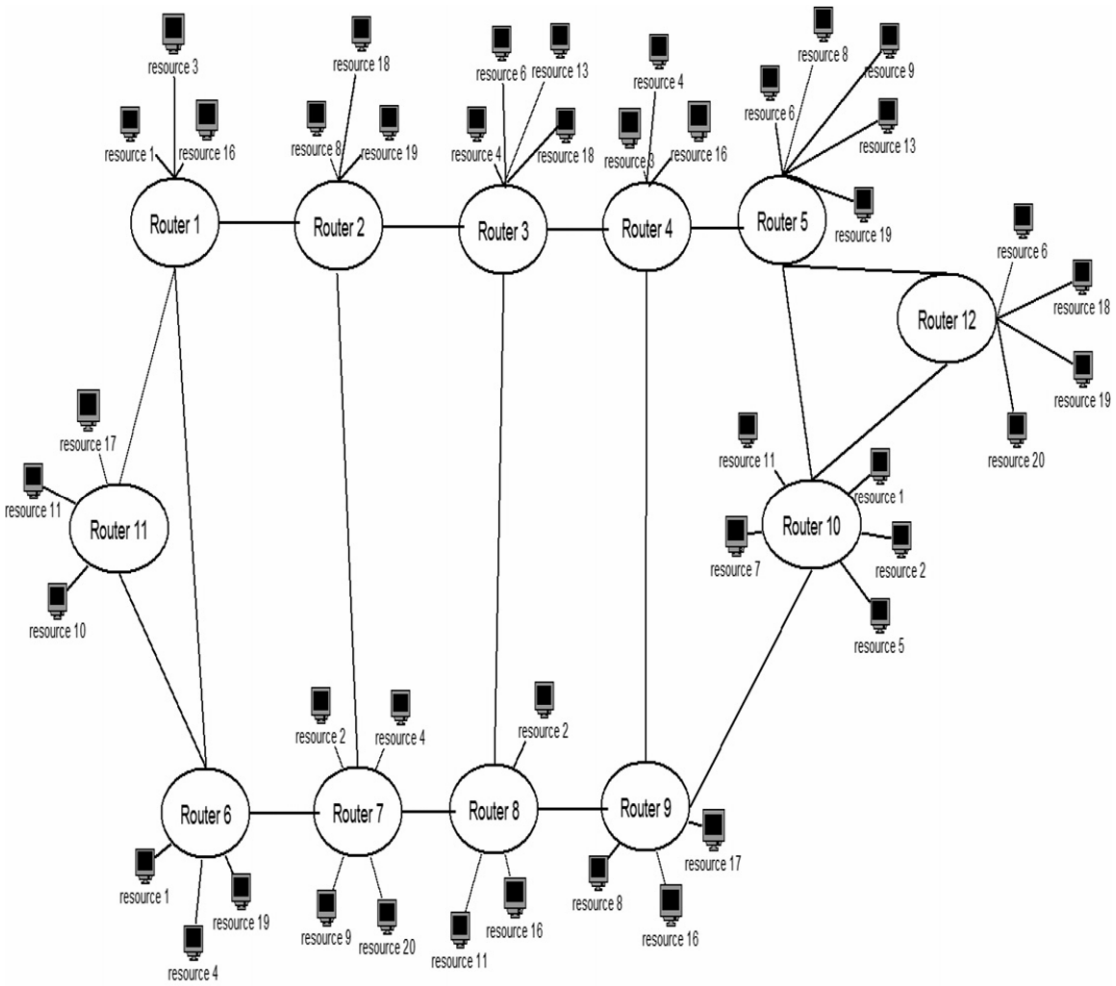
همانطوری که در این شکل ملاحظه می‌شود هر مسیریاب در محیط مسئول چندین منبع است و هر یک از مسیریاب‌های محیط نیز با چند مسیریاب دیگر در محیط در ارتباط است. به عنوان مثال همانطوری که در شکل ۲-۱ ملاحظه می‌کنید مسیریاب ۱۱ مسئول منابع ۱۰، ۱۱ و ۱۷ است و به طور مستقیم با مسیریاب های ۱و ۶ در ارتباط است.

---

[۲۱] Routing Table
[۲۲] Router

www.Prozhe.com

www.Prozhe.com



شکل۲-۱: نمونه‌ای از محیط گرید تشکیل شده از مسیریاب‌ها و منابع (Karaoglanoglou, ۲۰۰۸).

هر مسیریاب در محیط گرید یک جدول به نام جدول مسیریابی که اندازه آن برابر با انواع منابع موجود در گرید است، نگه می‌دارد. در هر یک از خانه‌های این جدول اعدادی نوشته شده است که نشان دهنده حداقل تعداد گام‌های اندازه‌گیری شده از مسیریاب مورد نظر تا تمامی منابع موجود و در دسترس در محیط است. نمونه‌ای از جدول مسیریابی را می‌توانید در شکل ۲-۲ ملاحظه کنید.

www.Prozhe.com

www.Prozhe.com

	Resource 1	Resource 2	...	...	Resource 20
Distance in hops from router to resource					

شکل۲-۲: نمونه‌ای از یک جدول مسیریابی (Karaoglanoglou, ۲۰۰۸).

با مشاهده شکل ۲-۲ متوجه می‌شویم از آنجایی که جداول مسیریابی کمترین فاصله هر مسیریاب را تا تمام منابع موجود در محیط نگه می‌دارند، در این روش یک الگوریتم کوتاه‌ترین مسیر، نقش اصلی را بازی می‌کند. در این روش، قبل از اینکه بتوانیم کشف منبع را انجام دهیم باید تمامی مسیریاب‌های موجود در محیط با اجرای الگوریتم کوتاه‌ترین مسیر، کمترین فاصله از تمامی منابع موجود در محیط را پیدا کرده و در جداول مسیریابی موجود در حافظه خود ذخیره کنند.

بعد از اینکه تمامی مسیریاب‌های محیط فاصله‌شان را از تمامی منابع موجود در محیط اندازه گرفته و در جداول خود ثبت کردند، یک کاربر می‌تواند درخواست کشف منبع خودش را به یکی از مسیریاب‌های موجود در محیط تحویل دهد. به عنوان مثالی از کشف منبع، همانطوری که در شکل ۲-۳ ملاحظه می‌کنید یک کاربر درخواستی را مبنی بر منبع ۱۸ به مسیریاب ۱۱ در محیط تحویل می‌دهد. این مسیریاب بلافاصله بررسی می‌کند که آیا منبع مورد درخواست کاربر را به صورت محلی در اختیار دارد یا خیر. چون این مسیریاب منبع درخواستی کاربر را به صورت محلی در اختیار ندارد جدول خود را برای فهمیدن اینکه درخواست را به کدامیک از همسایه‌هایش ارسال کند بررسی می‌کند و نهایتا براساس کوتاه‌ترین فاصله تا منبع تصمیم می‌گیرد که درخواست را به مسیریاب ۱ ارسال کند. مسیریاب ۱ نیز به همین ترتیب چون منبع مورد نیاز کاربر را به صورت محلی در اختیار ندارد با بررسی جدول خود درخواست را به مسیریاب ۲ تحویل می‌دهد. در اینجا مسیریاب ۲ منبع ۱۸ را به صورت محلی در اختیار دارد و آن را برای کاربر رزرو می‌کند.

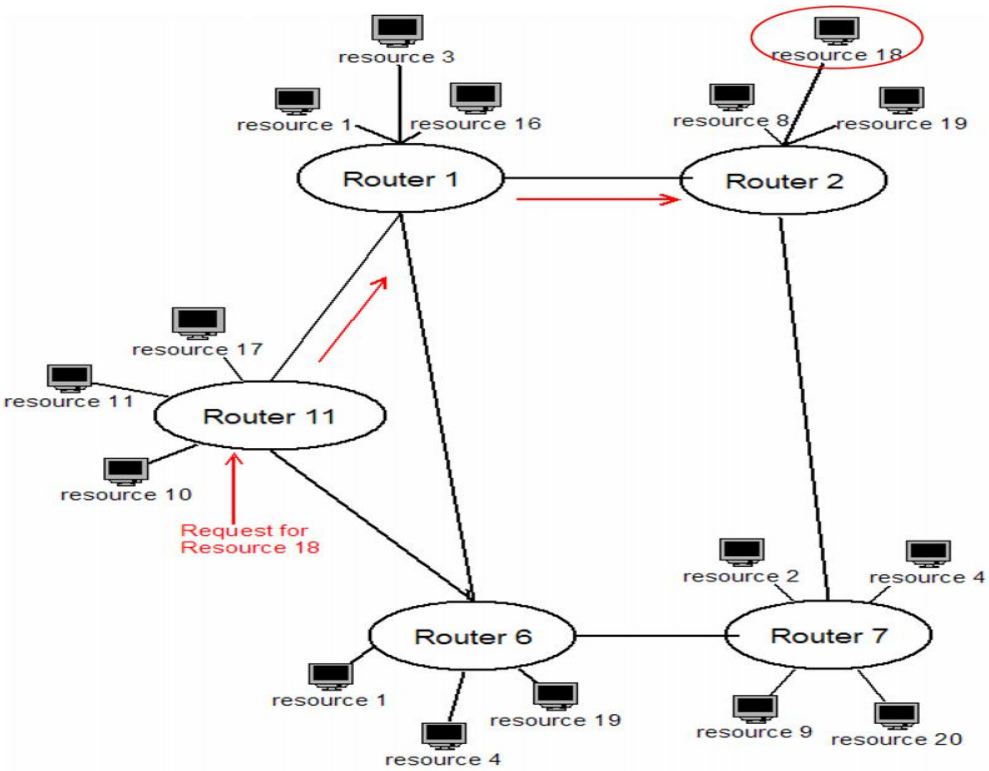
www.Prozhe.com

www.Prozhe.com

همانگونه که ملاحظه کردید در این روش برای کشف منبع از کوتاه‌ترین فاصله تا منبع که در جداول مسیریابی گره‌ها ذخیره شده است استفاده می‌شود. در این روش گره‌های موجود در محیط (مسیریاب‌ها) توسط ساختار گراف به یکدیگر متصل شده‌اند.

۲-۳- روش پیشنهادی Li برای کشف منبع در محیط گرید

در مرجع (Li, ۲۰۱۰)، روشی برای کشف منبع در گرید پیشنهاد شده است که کشف منبع را بین یکسری از گره‌های مشابه انجام می‌دهد. در این روش هر یک از گره‌های موجود در محیط یک سر خوشه[۲۳]



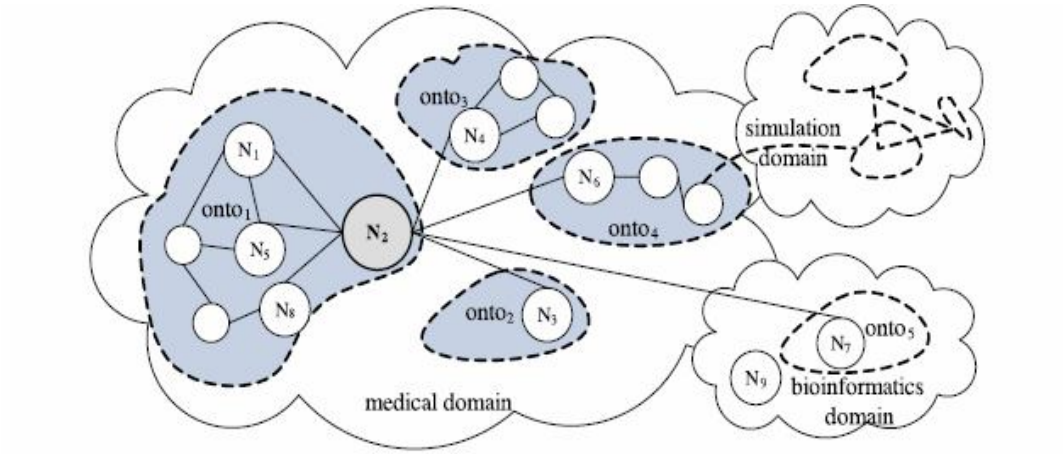
شکل۲-۳: مثالی از یک کشف منبع (Karaoglanoglou, ۲۰۰۸).

[۲۳] Cluster Head

www.Prozhe.com

www.Prozhe.com

هستند که مسئولیت یکسری از گره‌هایی که با یکدیگر مشابه هستند را بر عهـده دارنـد. نمونـه‌ای از این محیط در شکل ۲-۴ نشان داده شده است.



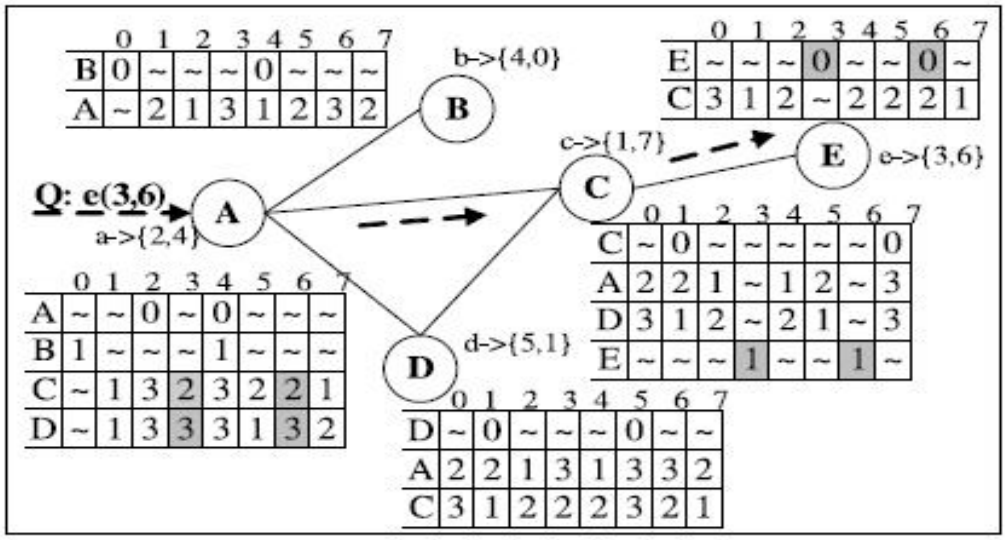
شکل۲-۴: نمونه‌ای از توپولوژی شبکه (Li, ۲۰۱۰).

کشف منبع در این روش در دو حالت یکی بین خوشه‌ای و دیگری درون خوشـه‌ای انجـام مـی‌شـود. برای کشف منبع در این روش در بین خوشه‌ها، درخواست‌هـای کـشف منبـع بـه صـورت گـام‌هـای تصادفی[۲۴] یا گام‌های سیل‌آسا [۲۵] فرستاده می‌شوند و برای کشف منبع درون خوشـه‌هـا از یکـسری جداول مسیریابی به نام RDVT استفاده می‌شود. در این جداول فاصله منابع تا گره‌ها نگهداری مـی-شود. نمونه‌ای از کشف منبع در شکل ۲-۵ نشان داده شده است.

[۲۴] Random- walk- based
[۲۵] Flooding- walk- based

www.Prozhe.com

www.Prozhe.com



شکل۲-۵: مثالی از کشف منبع (Li, ۲۰۱۰).

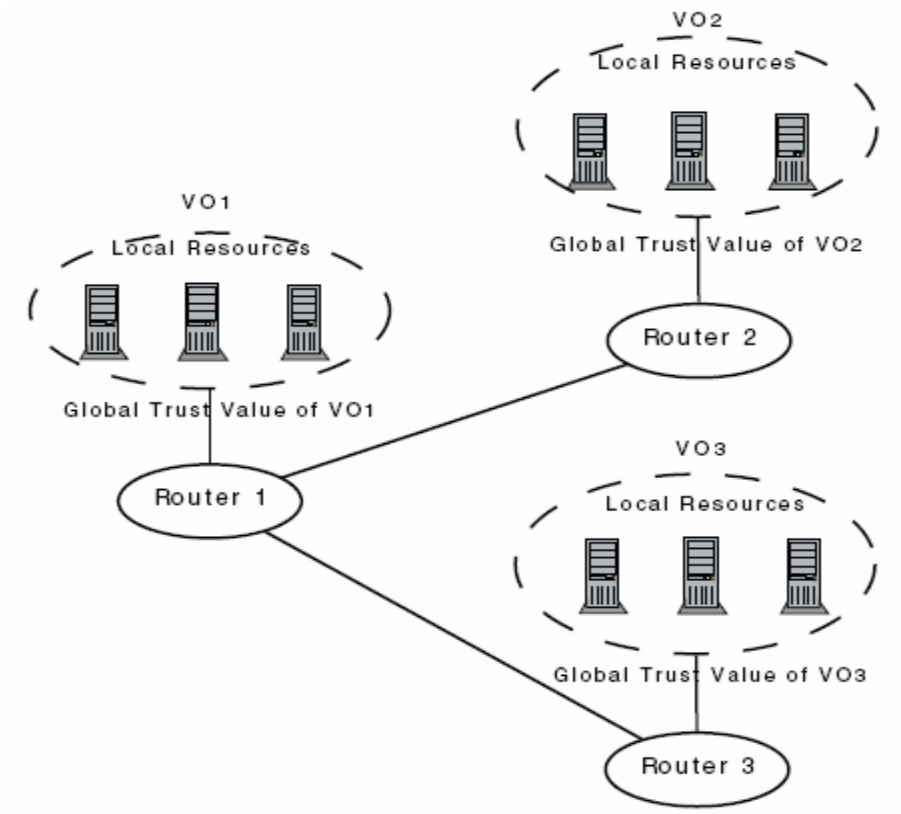
۲-۴- کشف منبع با استفاده از مقادیر اعتبار برای سازمان‌های مجازی

در کار پیشنهادی در مرجع (Karaoglanoglou, ۲۰۱۱)، یک روش کشف منبع پیشنهاد شده است که عمل کشف را در محیطی انجام می دهد که شامل مسیریاب‌ها و منابع است. در این روش هر مسیریاب مسئول یک سازمان مجازی[۲۶] است که شامل چندین منبع می‌باشد. به عبارت دیگر هر مسیریاب مجموعه‌ای از منابعی را که متعلق به یک سازمان مجازی است را کنترل می‌کند. در این روش همچنین یکسری از مقادیر معتبر[۲۷] به سازمان‌های مجازی اختصاص داده می‌شود. علت این کار نیز ارسال درخواست‌های کاربران به سازمان‌های مجازی معتبر و منابع در دسترس محیط است. نمونه‌ای از این محیط در شکل ۲-۶ آورده شده است.

[۲۶] Virtual Organization
[۲۷] Trust values

www.Prozhe.com

www.Prozhe.com



شکل۲-۶: نمونه‌ای از یک محیط گرید فرضی متشکل از سازمان‌های مجازی (Karaoglanoglou, ۲۰۱۱).

در این روش نیز وقتی درخواست کشف منبع به یکی از مسیریاب‌های محیط می‌رسد، آن مسیریاب با بررسی جدول مسیر خود درخواست را به مسیر صحیح هدایت می‌کند و اگر مقدار اعتبار یک سازمان مجازی از یک حد آستانه[۲۸] کمتر باشد، آن سازمان اجازه دسترسی به منابع خود را نمی‌دهد و منابع آن سازمان از لیست جداول مسیریاب‌های محیط پاک می‌شود. این کار باعث می‌شود که همواره به منابع معتبر و در دسترس مراجعه شود و درخواست کاربران به منابع خارج از دسترس به اشتباه ارسال نشود.

[۲۸] Threshold

www.Prozhe.com

www.Prozhe.com

۲-۵- روش کشف منبع با استفاده از نقشه بیتی در گرید

در مرجع (Chang, ۲۰۱۰)، نویسندگان یک روش کشف منبع با استفاده از نقشه بیتی بر روی یک ساختار درختی را پیشنهاد داده‌اند. آن‌ها گره‌های موجود در محیط را با ساختار درختی به یکدیگر متصل کرده‌اند و از دو نقشه بیتی به نام‌های "index bitmap" و"local resource bitmap" و یک نقشه بیتی اضافی به نام "attribute bitmap" استفاده کرده‌اند. اطلاعات مربوطه به منابع محلی در local resource bitmap ها ذخیره می‌شود و گره‌ها می‌توانند با بررسی این نقشه بیتی متوجه شوند که منبع درخواستی کاربر را در اختیار دارند یا نه. نقشه بیتی دیگر index bitmap است که وظیفه نگهداری اطلاعات منابع فرزندان و نسل‌های یک گره را بر عهداه دارد . این نقشه بیتی فقط در گره‌هایی وجود دارد که گره برگ[۲۹] نباشند. این نقشه بیتی از حاصل OR منطقی بین نقشه‌های بیتی فرزندان یک گره ایجاد می‌شود. در نهایت نقشه بیتی counter bitmap که تعداد هر کدام از منابع را نگه می‌دارد و در عمل به روز رسانی استفاده می‌شود. نمونه‌ای از این نقشه‌های بیتی نیز در شکل ۲-۶ مشخص است.

روش کشف منبع در این روش به این صورت است که ابتدا کاربر درخواست خود را به فرم یک نقشه بیتی به یکی از گره‌های موجود در محیط تحویل می‌دهد. آن گره ابتدا درخواست رسیده از کاربر را با نقشه بیتی محلی خود AND می‌کند. این کار برای این است که بررسی کند آیا منبع مورد درخواست کاربر در خود این گره به صورت محلی وجود دارد یا نه. اگر حاصل AND چیزی غیر از صفر شد، یعنی این گره منبع درخواستی را در اختیار دارد. این گره منبع را برای کاربر رزرو می‌کند و پاسخ موفقیت آمیزی را به سمت گره‌ای که درخواست را فرستاده بود، می‌فرستد. در حالت دوم حاصل AND صفر می‌شود پس این گره درخواست را با نقشه بیتی فرزندان یعنی index bitmap خود AND می‌کند. در این جا نیز دو حالت ممکن است وجود داشته باشد:

- ✓ در حالت اول حاصل AND صفر می‌شود که نشان می دهد منبع درخواستی کاربر در فرزندان و نسل‌های این گره (زیر شاخه‌ها) وجود ندارد. در این حالت گره درخواست را به سمت گره والد خود می‌فرستد.

---

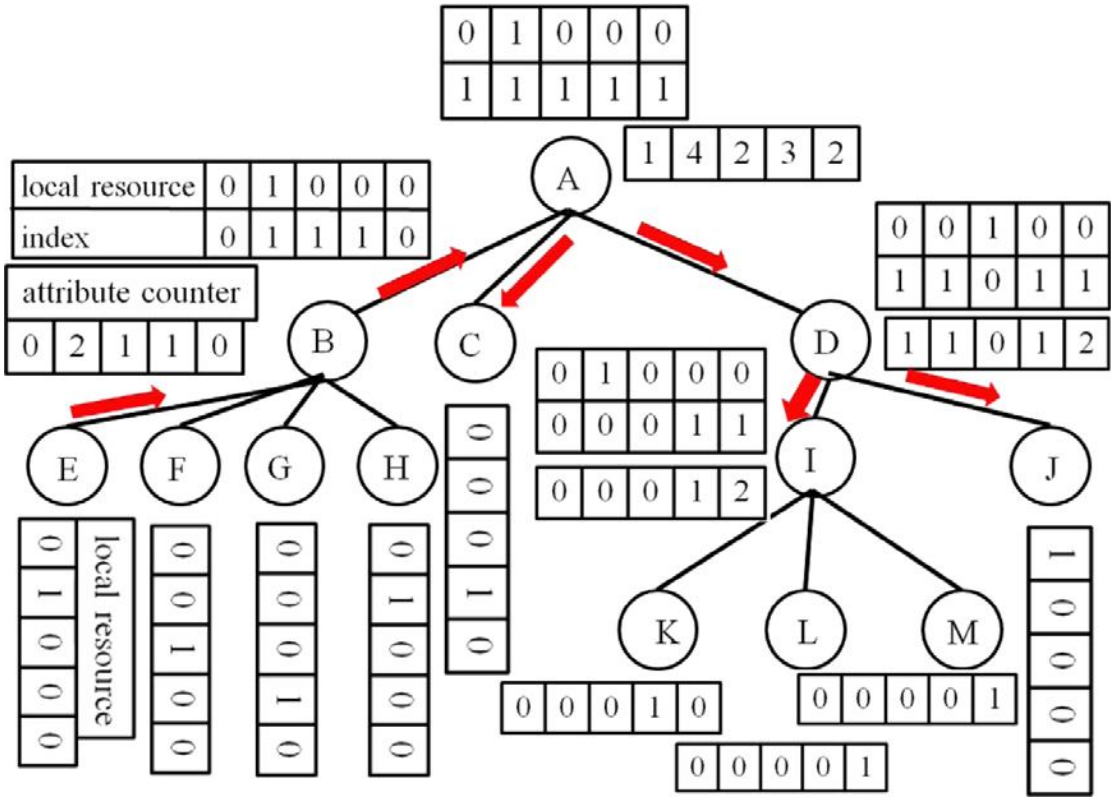
[۲۹] Leaf node

www.Prozhe.com

www.Prozhe.com

- ✓ در حالت دوم حاصل AND عددی غیر از صفر می‌شود که در این حالت گره درخواست کـاربر را بین فرزندان خود پخش می‌کند و در گره‌های بعدی نیز همین عملیات تکرار می- شود تـا در نهایت منبع مورد درخواست کاربر، اگر در محیط وجود داشته باشـد، بـرای کـاربر کـشف شود.

نمونه‌ای از کشف در این محیط نیز در شکل ۲-۷ مشاهده می‌شود.



شکل۲-۷: مثالی از کشف منبع بین گره‌های محیط (Chang, ۲۰۱۰).

همان‌طوری که ملاحظه می‌شود این روش توانسته است بسیاری از گره‌های ملاقـات شـده کـه در روش‌هـای قبلی نظیر سیل‌آسا به صورت اضافی و غیر ضروری ملاقات می‌شدند را با هرس کردن بخشی از درخت هنگام

www.Prozhe.com

www.Prozhe.com

کشف منبع کاهش دهد. این روش نسبت به روش‌های پیش‌تر از خود یک روش موثرتر است. نتایج نیز این امر را نشان می‌دهند (به مرجع (Chang, ۲۰۱۰) مراجعه شود).

۲-۶- سایر روش‌های کشف منبع

در سال‌های اخیر، بیشتر تمرکز محققین بر روی دو نوع از سیستم‌ها بوده است: سیستم‌های گرید و سیستم‌های نظیر به نظیر[۳۰] . این دو سیستم هر دو سیستم‌های اشتراک منابع هستند. بسیاری از الگوریتم-های کشف منبع روش‌های نظیر به نظیر را برای استفاده در گرید بهبود داده‌اند (Bharambe, ۲۰۰۴ Talia, ۲۰۰۶; Oppenheimer, ۲۰۰۴; Iamnitchi, ۲۰۰۲; Al-Dmour, ۲۰۰۵; Zerfiridis, ۲۰۰۳; Shen, ۲۰۰۹). روش‌های نظیر به نظیر غیر ساخت یافته اغلب به صورت سیل‌آسا عمل می‌کنند که این روش دارای سربار ارتباطاتی می‌باشد. اغلب روش‌های نظیر به نظیر ساخت یافته نیز از DHT برای این کار استفاده می‌کنند که از جمله آن‌ها می توان روش‌های CHORD (Stoica, ۲۰۰۱) و CAN (Ratnasamy, ۲۰۰۱) را نام برد. اما به علت وجود منابع پویا و تعداد به روزرسانی‌های زیاد در محیط گرید، DHT سربار[۳۱] را افزایش خواهد داد. اگر DHT برای کشف منبع در گرید استفاده شود، مقیاس پذیری[۳۲] را نیز محدود خواهد کرد. در زیر چند روش کشف منبع دیگر را به اختصار بررسی می‌کنیم.

در مرجع (Tangpongprasit, ۲۰۰۵)، نویسندگان یک الگوریتم ذخیره‌سازی برای پیدا کردن منبع مناسب در گرید پیشنهاد داده‌اند. در مسیر ارسال درخواست، ابتدا بررسی می‌شود که آیا منبع درخواستی به صورت محلی در گره وجود دارد یا نه. اگر منبع درخواستی در گره وجود داشته باشد، منابع ذخیره و رزرو می‌شوند و در مسیر برگشت یکی از منابع رزرو شده انتخاب می‌شود، در غیر این صورت این الگوریتم درخواست را به صورت تصادفی به گره‌های دیگر ارسال می‌کند.

در مرجع (Qi, ۲۰۰۶)، نویسندگان از یک جدول که اطلاعات منابع را ذخیره می‌کند برای پیدا کردن منابع درخواستی خود استفاده می‌کنند.

---

[۳۰] Peer-to-Peer
[۳۱] Overhead
[۳۲] Scalability

www.Prozhe.com

www.Prozhe.com

از روش های دیگر کشف منبع می‌توان به روش‌هایی اشاره کرد که از ارتباطات معنایی[۳۳] برای کشف منبع در محیط گرید استفاده می‌کنند (Li, ۲۰۰۵; Zhu, ۲۰۰۵; Nazir, ۲۰۰۵; Zhuge, ۲۰۰۴). نمونه‌ای از این روش‌ها نیز در مرجع (Li, ۲۰۱۰) آورده شده است که در بخش‌های قبلی این فصل معرفی شد. در این روش‌ها به گره‌های محیط این امکان داده می‌شود که بدون نیاز به یک نقطه ملاقات مرکزی با یکدیگر ارتباط داشته باشند. یکی دیگر از روش‌هایی که برای کشف منبع در گرید استفاده می‌شود، matchmaking (Raman, ۱۹۹۸) است. روش matchmaking برای حل مسائل موجود در conder طراحی شد. بر طبق این روش تمام موجودیت[۳۴] های موجود در گرید به دو گروه تقسیم می‌شوند: درخواست‌کنندگان[۳۵] و فراهم‌کنندگان[۳۶]. تمام موجودیت‌های مشخصات خود را در محیط اعلام می‌کنند و یک سرویس matchmaking مسئول یافتن یک مطابقت بین منابع و درخواست‌ها می‌باشد.

البته تاکنون روش‌های بسیار زیادی در زمینه کشف منبع در گرید به دلیل اهمیت این موضوع پیشنهاد شده است که ما در این فصل بخشی از این کارها را جهت آشنایی خوانندگان آوردیم. در بخش بعد تفاوت روش پیشنهادی خودمان با روش‌هایی که بررسی شدند را به صورت کلی بحث خواهیم کرد.

۲-۷- تفاوت روش پیشنهادی خودمان با روش‌های بحث شده

ما در این پایاننامه یک روش کشف منبع در گرید را پیشنهاد داده‌ایم که برای اولین بار در محیط گرید از یک ساختار درخت وزن‌دار استفاده می‌کند. در این روش ما از یک سری نقشه‌های بیتی برای نگهداری اطلاعات منابع محلی و همچنین اطلاعات منابع فرزندان و نسل‌های یک گره استفاده می‌کنیم. با استفاده از این درخت وزن‌دار ما قادر خواهیم بود درخواست کاربر را از طریق یک مسیر منحصر به فرد به گره مقصد هدایت کنیم. به طور کلی تفاوت‌های اساسی را که روش ما نسبت به روش‌های دیگر دارد را می توان در موارد زیر بیان کرد:

---

[۳۳] Semantic communities
[۳۴] Entity
[۳۵] Requestors
[۳۶] Providing

www.Prozhe.com

www.Prozhe.com

۱- الگوریتم ما از یک ساختار درخت وزن‌دار به منظور کشف منبع در گرید استفاده کرده است. مزیت روش ما این است که هر گره یک مسیر منحصر به فرد (یا یک آدرس منحصر به فرد) دارد. این مسیر منحصر به فرد از طریق وزن‌های موجود در درخت امکان پذیر است. بنابراین درخواست کاربر برخلاف تمام روش‌های قبلی از طریق یک مسیر مستقیم و بدون مراجعه به گره‌های اضافی و غیر ضروری به مقصد هدایت می‌شود. ما می‌توانیم مسیر مستقیم رسیدن به گره هدف را با استفاده از ردپاهایی که در نقشه‌های بیتی موجود در گره‌ها ذخیره می‌شود، پیدا کنیم.

۲- برای آماده‌سازی درخت و کشف منبع، تنها از یک نقشه بیتی در گره‌ها استفاده می‌کنیم. هر گره برای ذخیره اطلاعات مربوط به منابع محلی و همچنین اطلاعات مربوط به گره‌های فرزندان و نسل‌هایش تنها از این نقشه بیتی استفاده می کند.

روش پیشنهادی ما به دلیل اینکه از طریق یک مسیر مستقیم گره هدف را کشف می‌کند، هزینه کشف بسیار کمی داشته و نسبت به سایر روش‌هایی که اخیرا ارائه شده‌اند بهینه‌تر است. در فصل مربوط به شبیه‌سازی‌ها این مطلب نشان داده می‌شود.

۲-۸- نتیجه گیری

همانطوری که در این فصل مشاهده کردید تاکنون روش‌های بسیاری برای کشف منبع در محیط گرید پیشنهاد شده‌اند. تمامی این روش‌ها سعی دارند که با کمترین هزینه منبع مورد درخواست کاربر را کشف کرده و محیط را به روز رسانی کنند. هر روشی نیز که در مرحله مشف منبع هزینه کمتری از لحاظ تعداد گره‌های ملاقات شده و ترافیک تولیدی داشته باشد، می‌توان گفت که یک روش بهینه‌تر و قابل استفاده در محیط واقعی گرید است. به طور کلی هیچکدام از روش‌هایی که تاکنون ارائه شده‌اند نتوانسته‌اند بر روی ساختار درختی منبع مورد درخواست کاربر را بدون مراجعه به گره‌های غیر ضروری و اضافی و ایجاد ترافیک اضافی برای کاربر کشف کنند. این امر به دلیل این است که گره‌ها اطلاعات کافی از محیط را در اختیار ندارند و مجبورند در برخی از مقاطع از روش همه‌پخشی استفاده کنند. با اینکه روش‌هایی هم که اخیرا بر روی ساختار درختی ارائه شده‌اند توانسته‌اند یک سری از مراجعات غیر ضروری را به گره‌ها در گرید کاهش دهند ولی خود این روش‌ها نیز در بعضی از حالات مجبورند برای کشف منبع درخواستی کاربر از روش پخش

www.Prozhe.com

www.Prozhe.com

درخواست استفاده نمایند. این تعداد ملاقات گره‌های اضافی اگرچه ممکن است در برخی از محیط‌ها قابل اغماض باشد ولی در محیطی مانند گرید که تعداد گره‌های موجود در آن حتی ممکن است به چندین میلیون برسد و کاربران آن نیز بسیار زیادند، این تعداد گره‌های اضافی ملاقات شده بار سنگینی را به سیستم تحمیل خواهند کرد و از قابلیت و مقیاس‌پذیری سیستم به شدت خواهند کاست. روشی را هم که ما در این پایاننامه ارائه کرده‌ایم، سعی بر آن دارد تا آنجا که ممکن است از ملاقات گره‌های غیر ضروری کاسته و ترافیک را در سیستم پایین آورد. ما با ارائه یک ساختار درخت وزن‌دار و روشی که برای کشف منبع ارائه داده‌ایم موفق شدیم به چنین هدفی دست پیدا کنیم. روش کشف منبع ما می‌تواند منبع مورد درخواست کاربر را با ملاقات حداقل تعداد گره‌ها از طریق یک مسیر مستقیم کشف کند. نتایج شبیه‌سازی‌ها نیز این واقعیت را به خوبی نشان می‌دهند. در فصل بعد روش پیشنهادی خودمان را به طور مفصل توضیح خواهیم داد و در فصل‌های بعدی ارزیابی و مقایسه‌ای از روش پیشنهادی خودمان را نسبت به روش‌های دیگر خواهیم آورد.

www.Prozhe.com

www.Prozhe.com

فصل سوم

www.Prozhe.com

www.Prozhe.com

ارائه الگوریتم پیشنهادی

۳-۱- مقدمه

همانگونه که قبلاً بحث شد الگوریتم‌های کشف منبعی که برای محیطی مانند گرید، که تعداد منابع و درخواست‌ها بسیار زیاد است، پیشنهاد می‌شوند باید فاکتورهایی مانند هزینه بسیار پایین بر کشف منبع و به روز رسانی، ترافیک تولیدی بسیار کم در مراحل آماده‌سازی، کشف و به روز رسانی، مقیاس‌پذیری، سادگی روش بدون نیاز به محاسبات پیچیده، فضای مورد نیاز کم برای ذخیره اطلاعات (هر چقدر اطلاعات کمتر و منظم‌تر باشند مدیریت آنها آسان‌تر است)، را در نظر بگیرند.

در روشی که در این پایاننامه بر روی آن کار کرده‌ایم تمامی فاکتورهای بالا را در نظر گرفته‌ایم. روش کشف منبع ما دارای هزینه کشف و به روز رسانی کم و ترافیک تولیدی پایین بوده و نیز بهبودهایی را در روش پیشنهادی‌مان داده‌ایم تا روش پیشنهادی کشف منبع ما مقیاس‌پذیر باشد. عملیاتی هم که در مراحل کشف منبع یا به روز رسانی استفاده می‌شوند، از عملگرهای ساده‌ای مانند AND، ضرب و .... استفاده می‌کنند و پیچیدگی مکانی[۳۷] الگوریتم کشف منبع ما نیز بیشتر از سایر روش‌ها نیست. در بخش‌های بعدی این فصل ابتدا نقشه‌های بیتی را که در روش استفاده می‌شوند را توضیح خواهیم داد، سپس نحوه تشکیل درخت و فاز آماده‌سازی قبل از کشف منبع را توضیح می‌دهیم. در بخش‌های بعدی روش کشف منبع و به روز رسانی را توضیح خواهیم داد و سپس راهکارهایی را برای بهبود روش خود ارائه می‌دهیم، سپس بخشی با عنوان پیچیدگی را برای بررسی پیچیدگی زمانی[۳۸] و مکانی آوردیم و در نهایت خلاصه‌ای از فصل را خواهیم آورد.

۳-۲- توصیف نقشه های بیتی استفاده شده در الگوریتم

به دلیل اینکه در محیط گرید منابع مختلفی وجود دارند و هر منبع نیز صفت‌های متفاوتی دارد بنابراین باید این منابع و صفت هایشان به گونه‌ای در بسته‌های پیشنهادی ما مشخص باشند. ما در روش پیشنهادی

---

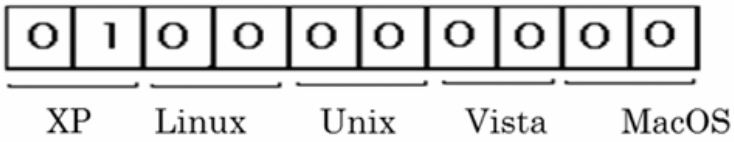
[۳۷] Space complexity
[۳۸] Time complexity

www.Prozhe.com

www.Prozhe.com

خودمان از دو نوع نقشه‌بیتی به نام‌های "نقشه بیتی مسیر"[۳۹] و "نقشه بیتی شمارنده"[۴۰] استفاده خواهیم کرد. در ابتدا، ما نقشه بیتی مسیر را توضیح می‌دهیم.

۱- این نقشه بیتی دارای خانه‌هایی است که اعدادی در آنها ذخیره خواهند شد. طول این نقشه بیتی به تعداد صفت‌های منبع بستگی خواهد داشت. به عنوان مثال اگر منبع مورد نظر ما یک سیستم عامل باشد و انواع سیستم عامل‌هایی که ما در محیط خود به اشتراک خواهیم گذاشت عبارت باشند از: Macos، Vista، XP، Unix و Linux آنگاه ما در نقشه بیتی خود ۱۰ خانه خواهیم داشت. به عبارت دیگر برای هر نوع منبع دو خانه از نقشه بیتی اختصاص داده می‌شود: شمارنده و ردپا. تمامی درخواست های کاربران نیز باید در قالب این نقشه بیتی بوده و به یکی از گره‌های موجود در محیط تحویل داده می‌شود. شکل ۳-۱ نمونه‌ای از این نقشه‌بیتی را نشان می‌دهد که شامل سیستم عامل XP در گره مربوطه است.



شکل۳-۱: نمونه ای از یک نقشه بیتی مربوط به سیستم عامل (Khanli, ۲۰۱۱).

این اعداد به این صورت در مکان‌های مربوطه قرار می‌گیرند: چون منبع XP مربوط به خود گره است و آن را به صورت محلی در اختیار دارد و از فرزند خود دریافت نکرده است، در قسمت شمارنده عدد ۰ قرار می‌دهد و وجود عدد ۱ هم نشان‌دهنده این است که این منبع در این گره وجود دارد. خانه‌های دیگر ۰ خواهد بود و نشان‌دهنده عدم وجود منبع در خود گره یا فرزندان و نسل‌های آن گره می‌باشد. هر گره در درخت کشف منبع ما، در ابتدا چنین نقشه بیتی را خواهد داشت و در مراحل بعد محتوای این خانه ها تغییر خواهد کرد.

۲- نقشه بیتی دیگر نقشه بیتی شمارنده است. تعداد خانه‌های این نقشه بیتی به انواع منابع موجود در محیط گرید بستگی داشته و با آن برابر است. برای مثال قبلی چون انواع سیستم عامل‌های موجود در محیط ۵ تا بود بنابراین این نقشه بیتی ۵ خانه خواهد داشت. این نقشه‌بیتی در هر یک از گره‌های موجود در محیط

[۳۹] Path bitmap
[۴۰] Counter bitmap

www.Prozhe.com

www.Prozhe.com

تعداد منابع موجود در فرزندان یا نسل‌های آن گره را نمایش خواهد داد و کاربرد اصلی آن نیز در مسئله به روز رسانی است.
الگوریتم پیشنهادی ما بر روی یک درخت وزن‌دار است. در این معماری، وزن یال‌های درخت توسط گره‌های والد اختصاص داده خواهند شد. یعنی هر گره بسته به تعداد فرزندانی که در اختیار می‌گیرد به هر یک از فرزندان یک عدد اختصاص می‌دهد. این گره می‌تواند طول وزن‌های اختصاصی خود را از طریق فرمول ساده زیر محاسبه کند:

$$y = \lceil \log_2^n \rceil \qquad (۱-۳)$$

در این فرمول n تعداد فرزندان هر گره بوده و y حداقل تعداد بیت‌هایی است که آن گره می‌تواند به یال‌های فرزندان خود به عنوان وزن اختصاص دهد.
به این ترتیب ملاحظه می‌کنید که گره‌های موجود در محیط به راحتی ارتباطات خود را به همدیگر مشخص کرده و نقشه‌های بیتی خود را تشکیل داده و اطلاعات منابع محلی خودشان را ذخیره می‌کنند و منتظر مرحله آماده‌سازی درخت می‌مانند تا اطلاعات گره‌ها با یکدیگر مبادله شده و ساختار کلی درخت قبل از کشف منبع مشخص شود.

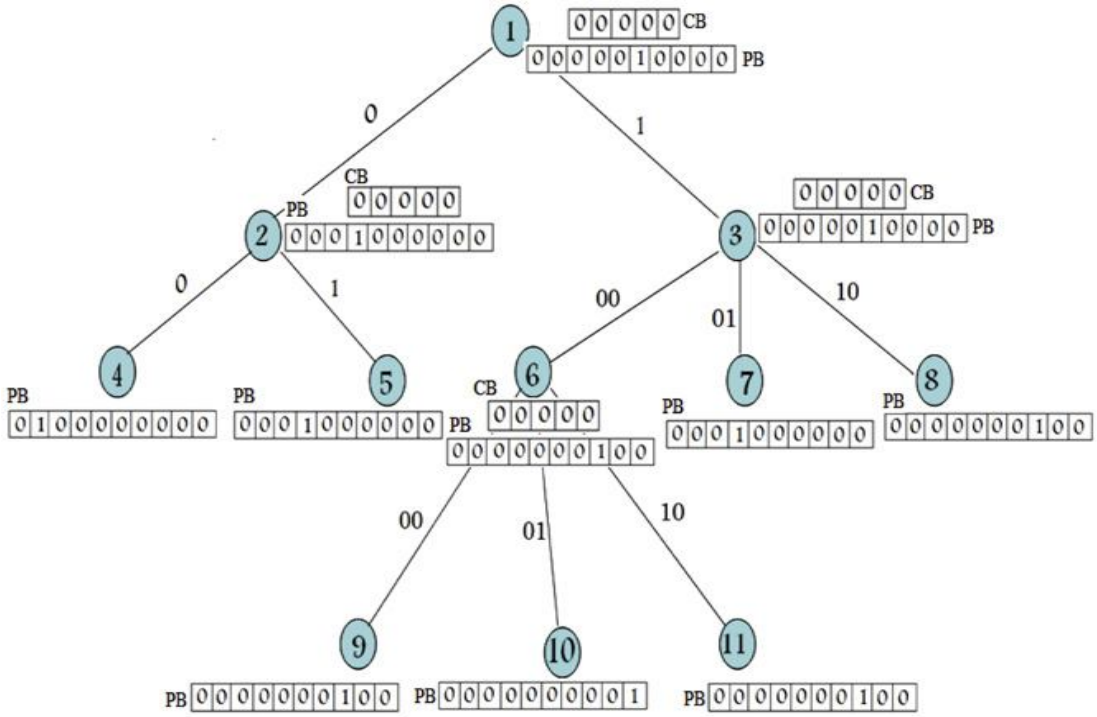
۳-۳- تشکیل درخت و توصیف الگوریتم

در شکل ۳-۲ نمونه‌ای از یک محیط گرید با ۱۱ گره نشان داده شده است . همانگونه که ملاحظه می‌شود گره‌ها اطلاعات منابع محلی خودشان را در نقشه‌های بیتی مربوطه ثبت کرده‌اند. این شکل قبل از مرحله آماده‌سازی درخت با منابع محلی گره‌ها بدون در نظر گرفتن منابع فرزندان است. شکل ۳-۶ همان درخت بعد از جمع آوری اطلاعات فرزندان توسط گره‌های والد و تشکیل فرم نهایی درخت کشف منبع وزن‌دار است. نحوه تشکیل درخت و فرم نهایی را با یک مثال شرح می‌دهیم.

www.Prozhe.com

www.Prozhe.com

فرض کنید منبع مورد نظر ما سیستم عامل باشد. همانطوری که در شکل ۳-۱ ملاحظه می‌شود هر جفت خانه نشان‌دهنده یک نوع سیستم عامل می‌باشد. به عنوان مثال در شکل ۳-۲، گره ۱۰ سیستم عامل Macos، گره ۳ سیستم عاملUnix، گره ۲ سیستم عامل Linux و غیره را در اختیار دارند. وزن یال‌های درخت بر روی آن‌ها نوشته شده است. همانطوری که گفته شد این وزن‌ها مثلا ۰ بین گره‌های ۱ و ۲ عددی است که گره ۱ به عنوان والد، در حافظه‌اش برای گره ۳ در نظر گرفته است. حال این گره‌ها باید فرم نهایی خود را تشکیل دهند تا یک کاربر بتواند درخواست خود را به یکی از این گره‌ها تحویل دهد و این درخت کشف منبع، منبع مورد درخواست کاربر را پیدا کند.



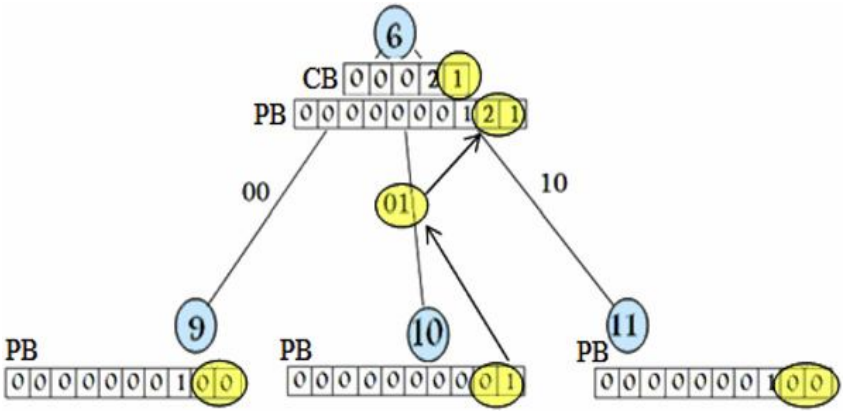
شکل۳-۲: نمونه‌ای از درخت کشف منبع ردپا قبل از مرحله آماده‌سازی (Khanli, ۲۰۱۱).

ابتدا گره‌های ۹ ، ۱۰ و ۱۱ را در نظر بگیرید. گره ۶ به عنوان گره والد، نقشه‌های بیتی این سه فرزند را گرفته و تجزیه و تحلیل می‌کند و اطلاعات نهایی را بر روی نقشه‌بیتی خود ذخیره می‌کند. فرض کنید گره ۶

www.Prozhe.com

www.Prozhe.com

اطلاعات گره‌های ۹، ۱۰ و ۱۱ را گرفت. ابتدا از خانه‌های پایین‌تر شروع می‌کنیم. دو خانه اول گره‌های ۹، ۱۰ و ۱۱ برداشته می‌شود. دوتای آنها محتوی ۰۰ و یکی از آنها محتوی ۰۱ است. محتوای دو خانه مربوطه خود گره هم ۰۰ است. بنابراین تنها یک منبع و آن هم در گره ۱۰ وجود دارد که باید در موقعیت مربوطه گره ۶ ثبت شود. چون این منبع از مسیری با وزن ۰۱ آمده است در خانه اول، مسیری که منبع از آنجا آمده یعنی ۰۱ نوشته می‌شود و خانه دوم هم که شمارنده را نشان خواهد داد دو واحد اضافه می‌شود که در نهایت عدد ۲۱ در مکان‌های مربوطه در نقشه بیتی گره ۶ ثبت می‌شود. خانه اول نقشه بیتی شمارنده هم در گره ۶ بدلیل اینکه تنها یک منبع از فرزندان خود دریافت کرد عدد ۱ نوشته می‌شود. در شکل ۳-۳ این فرآیند نشان داده شده است. حال دو خانه بعدی گره‌های ۹ ، ۱۰ و ۱۱ را در نظر بگیرید. دو خانه مربوط به گره ۱۰، ۰۰ است و



شکل۳-۳: جمع آوری اطلاعات فرزندان توسط گره ۶ (مربوط به MacOS) (Khanli, ۲۰۱۱).

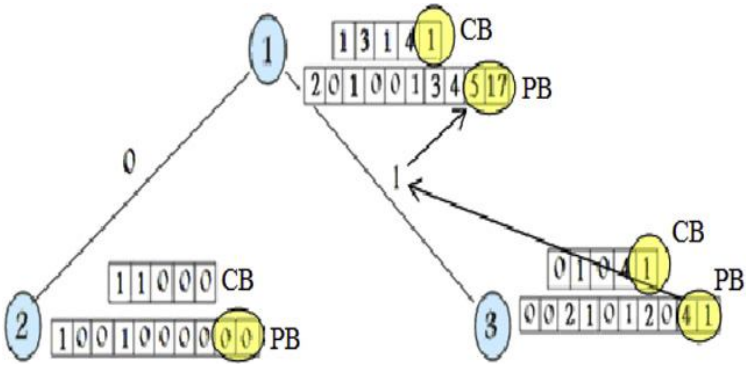
برای گره‌های ۹ و ۱۱ هم ۰۱ می‌باشد. این به معنی آن است که گره‌های ۹ و ۱۱ این منبع را در اختیار دارند و گره ۶ نیز اطلاعات یکی از آنها را در موقعیت مربوطه نقشه‌بیتی خود ثبت می‌کند، ولی با مراجعه به نقشه بیتی خود متوجه می‌شود که خودش این منبع را به صورت محلی در اختیار دارد و دیگر نیازی نیست که اطلاعات مربوط به منبعی را که در اختیار فرزندانش است را جایگزین اطلاعات خود بکند. بنابراین این دو خانه به همان صورت ۰۱ باقی می‌مانند و خانه مربوطه در نقشه بیتی شمارنده هم دو واحد افزایش پیدا می-

www.Prozhe.com

www.Prozhe.com

کند. بقیه خانه‌های گره ۶ هم چون هیچ اطلاعاتی از فرزندان خود دریافت نکرده به همان صورت باقی می‌ماند.

در یک مثال دیگر، فرض کنید گره ۱ اطلاعات دو گره ۲ و۳ را دریافت کرد. در اولین موقعیت نقشه بیتی، چون تنها گره ۳ حاوی اطلاعات منبع است، ۱ دهدهی[۴۱] به ۱ باینری[۴۲] به طول ۴ تبدیل می‌شود و به حاصل، وزن یال مربوطه اضافه می‌شود و در نهایت از حالت دهدهی به دودویی تغییر یافته و در موقعیت مربوطه قرار می‌گیرد. نقشه بیتی شمارنده هم یک واحد افزایش می‌یابد. تمامی این روند را می‌توانید در شکل ۳-۴ ملاحظه کنید.



شکل۳-۴: جمع‌آوری اطلاعات فرزندان توسط گره ۱ (مربوط به MacOS) (Khanli, ۲۰۱۱).

روند جمع‌آوری و تکمیل اطلاعات نقشه‌های بیتی مسیر و شمارنده برای درک بیشتر توسط الگوریتم شکل ۳-۵ آورده شده است. همان‌طوری که در این شکل ملاحظه می‌کنید الگوریتم از دو قسمت الگوریتم نحوه تشکیل نقشه‌بیتی مسیر و الگوریتم نحوه تشکیل نقشه‌بیتی شمارنده تشکیل شده است.

[۴۱] Decimal
[۴۲] Binary

www.Prozhe.com

www.Prozhe.com

```
/*Algorithm of constructing of Path Bitmap (PB) */
1: Wait for PBs from all children
2: for each pair positions in PBs do
3:    if ((PB_L[J] , PB_L[J-1]) ≠ (0,1)) then
4:       w:= i that ((PB_i[J] less or equal than other PB[J]s) and (PB_i[J],PB_i[J-1])≠(0,0))
          /* we must find the pair positions that contain a resource and related counter less
         or equal than others*/
5:       PB_T[J-1]:=(The weight of edge of node i) attach (transform PB_w[J-1] to binary to
         length PB_w[J] ) /* if PB_w[J]=0 then PB_T[J-1]:= The weight of edge of node i*/
6:       PB_T[J]:=PB_w[J]+length of weight of edge of node i
7:       PB_L[J-1]:=Transform PB_T[J-1] to decimal
8:       PB_L[J]:=PB_T[J]

/*Algorithm of constructing of Counter Bitmap(CB)*/
1: Wait for PBs from all children
2: for each pair positions in PBs do
3:    k:=0
4:      for each child node i do
5:        if find local resource information then
6:           k:=k+1
7:        k:=k+CB_i[related position]
8:      CB_L[related position]:=k
```

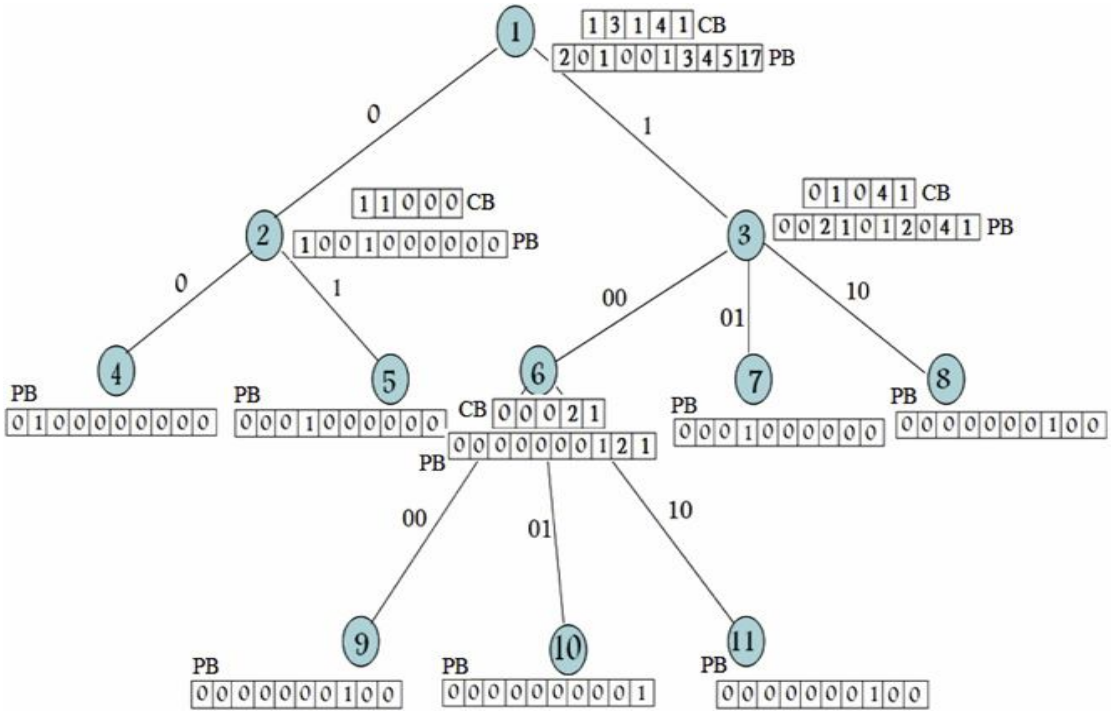
شکل۳-۵: الگوریتم تشکیل نقشه‌های بیتی مسیر و شمارنده (Khanli, ۲۰۱۱).

با استفاده از این الگوریتم‌ها به راحتی می‌توان اطلاعات منابع را در هر درخت فرضی جمع آوری کرد. تنها کافی است این الگوریتم‌ها در تمامی گره‌ها (به جز گره برگ) به صورت مستقل اجرا شوند. بعد از اجرای این الگوریتم‌ها توسط گره‌های موجود در محیط فرم نهایی درخت را می‌توان تشکیل داده و منبع مورد نظر توسط الگوریتم کشف منبع برای کاربر کشف شود. شکل ۳-۶ همان درخت وزن‌دار موجود در شکل ۳-۲ بعد از عبور از فاز آماده‌سازی است. همانگونه که در شکل ۳-۶ ملاحظه می‌کنید این درخت اکنون فرم نهایی محیط قبل از شروع کشف منبع است. اطلاعات منابع موجود در محیط در اختیار گره‌های والدشان قرار گرفته است و محل هر یک از منابع نیز برای گره‌های موجود در محیط مشخص است. اگر درخواستی مبنی

www.Prozhe.com

www.Prozhe.com

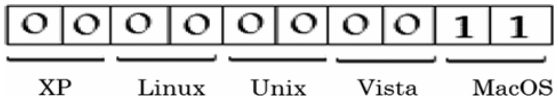
بر کشف هر یک از گره‌های محیط برسد، این گره‌ها می‌توانند با استفاده از اطلاعات ذخیره‌شده در نقشه‌های بیتی مسیر خود و با استفاده از الگوریتم کشف منبع درخواست کاربر را به سمت هدف هدایت کنند.



شکل۳-۶: نمونه‌ای از درخت کشف منبع ردپا بعد از مرحله آماده‌سازی (Khanli, ۲۰۱۱).

۳-۴- الگوریتم کشف منبع پیشنهادی (FRDT)

هنگامی که یک کاربر بخواهد درخواست کشف منبع خود را به یکی از گره‌های موجود در محیط تحویل دهد آن درخواست باید در یک قالب مشخص باشد. این قالب شبیه قالب‌هایی است که در گره‌ها ذخیره شده است. یعنی در مثال قبلی درخواست کاربر به صورت یک نقشه‌بیتی به طول ۱۰ خانه خواهد بود. نمونه‌ای از این درخواست در شکل ۳-۷ نشان داده شده است.



شکل۳-۷: نمونه‌ای از یک فرم درخواست (Khanli, ۲۰۱۱).

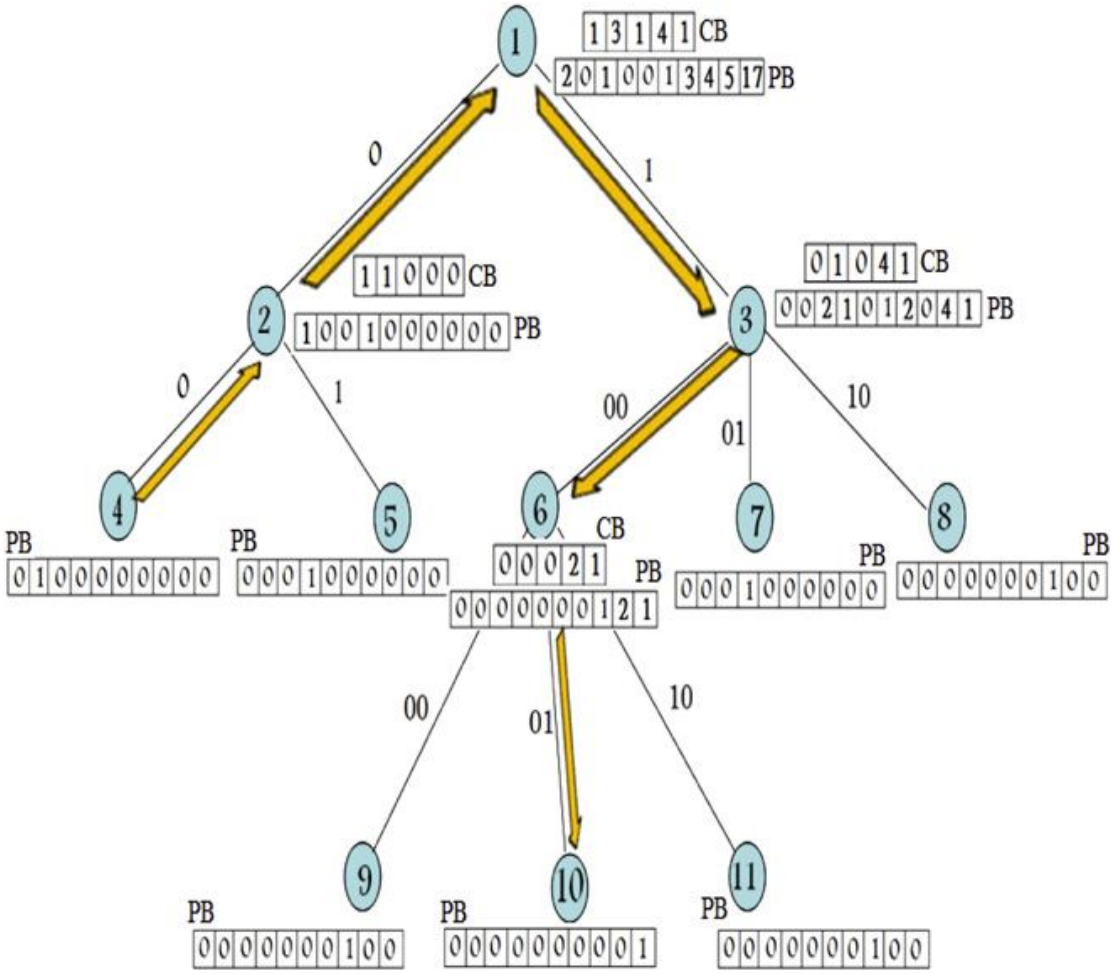
www.Prozhe.com

www.Prozhe.com

کاربر باید دو عدد ۱ در دو خانه مربوط به منبع مورد نظر خود قرار بدهد. اگر کاربر این درخواست را به یکی از گره‌های محیط تحویل دهد، الگوریتم کشف منبع در یک زمان کوتاه منبع مورد نظر کاربر را (اگر در محیط وجود داشته باشد) برای او کشف می‌کند. نمونه‌ای از یک کشف منبع در شکل ۳-۸ نشان داده شده است. فرض کنید یک کاربر به سیستم عامل MacOS احتیاج دارد. کاربر فرمی شبیه به آنچه که در شکل ۳-۷ نشان داده شده تشکیل خواهد داد. روش به این صورت است که هر گره تمامی خانه‌های درخواست را با خانه‌های نقشه‌های بیتی مسیر خود خانه به خانه ضرب می‌کند. اگر حاصل صفر شد، درخواست به سمت گره والد فرستاده می‌شود در غیر این صورت حاصل عددی خواهد بود که گره می‌تواند با استفاده از آن درخواست را مستقیما به گره‌ای که این منبع را در اختیار دارد از طریق یک مسیر مستقیم هدایت کند. در مثال نشان داده شده در شکل ۳-۸، درخواست ابتدا به گره ۴ تحویل داده می‌شود. گره ۴ خانه‌های درخواست را با خانه‌های موجود در حافظه خود خانه به خانه ضرب می‌کند و چون حاصل صفر می‌شود، گره ۴ درخواست را به والد خود یعنی گره ۲ تحویل می‌دهد. این گره نیز عملیاتی مشابه به آنچه که گره ۴ انجام داد، می‌دهد و درخواست را به گره والد خود یعنی گره ۱ ارسال می‌کند. این گره بعد از دریافت درخواست، تک تک خانه های آن را با خانه‌های نقشه بیتی خود ضرب می‌کند و به عدد ۱۷ ۵ دست پیدا می‌کند. این عدد آدرس رسیدن به مقصد است. این گره ۱۷ را از دهدهی به باینری به طول ۵ تبدیل کرده (۱۰۰۰۱) و آن را از سمت چپ به راست پیمایش می‌کند. اگر به ترتیب این اعداد بر روی یال‌های درخت با توجه به وزن‌ها پیش برویم ابتدا درخواست به گره ۳ ارسال خواهد شد و سپس به گره ۶ و در نهایت به گره ۱۰ خواهد رسید. گره ۱۰ منبع مورد درخواست کاربر را در اختیار دارد. به محض رسیدن درخواست کاربر، منبع را برای کاربر رزرو کرده و پاسخ موفقیت‌آمیزی را به سمت گره مبدا ارسال می‌کند. در شکل ۳-۸ همین جستجو نشان داده شده است که در نهایت منجر به کشف منبع در گره ۱۰ می‌شود. همانطوری که ملاحظه شد در این روش به محض اینکه به عددی به غیر از صفر رسیدیم با استفاده از این عدد و با استفاده از وزن-های درخت می‌توانیم به طور مستقیم و بدون مراجعه به هیچ گره اضافه‌ای به گره مقصد دست پیدا کنیم.

www.Prozhe.com

www.Prozhe.com



شکل۳-۸: نمونه‌ای از فرآیند کشف منبع در درخت کشف منبع وزن‌دار (Khanli, ۲۰۱۱).

۳-۵- الگوریتم به روز رسانی

روش کشف منبع پیشنهادی به این صورت بود که کاربر در دو خانه‌ای که درخواست آن منبع را دارد دو عدد ۱ قرار می‌دهد و درخواست کاربر به هر گره‌ای که می‌رسد خانه به خانه با نقشه بیتی گره مورد نظر ضرب می‌شود تا بالاخره به گره‌ای برسد که حاصل ضرب خانه‌ها عددی غیر از ۰ باشد، سپس از طریق این دو عدد و وزن یال‌ها مقصد را پیدا می‌کند. روش آپدیت به این صورت است که هنگامی که یک درخواست به گره‌ای

www.Prozhe.com

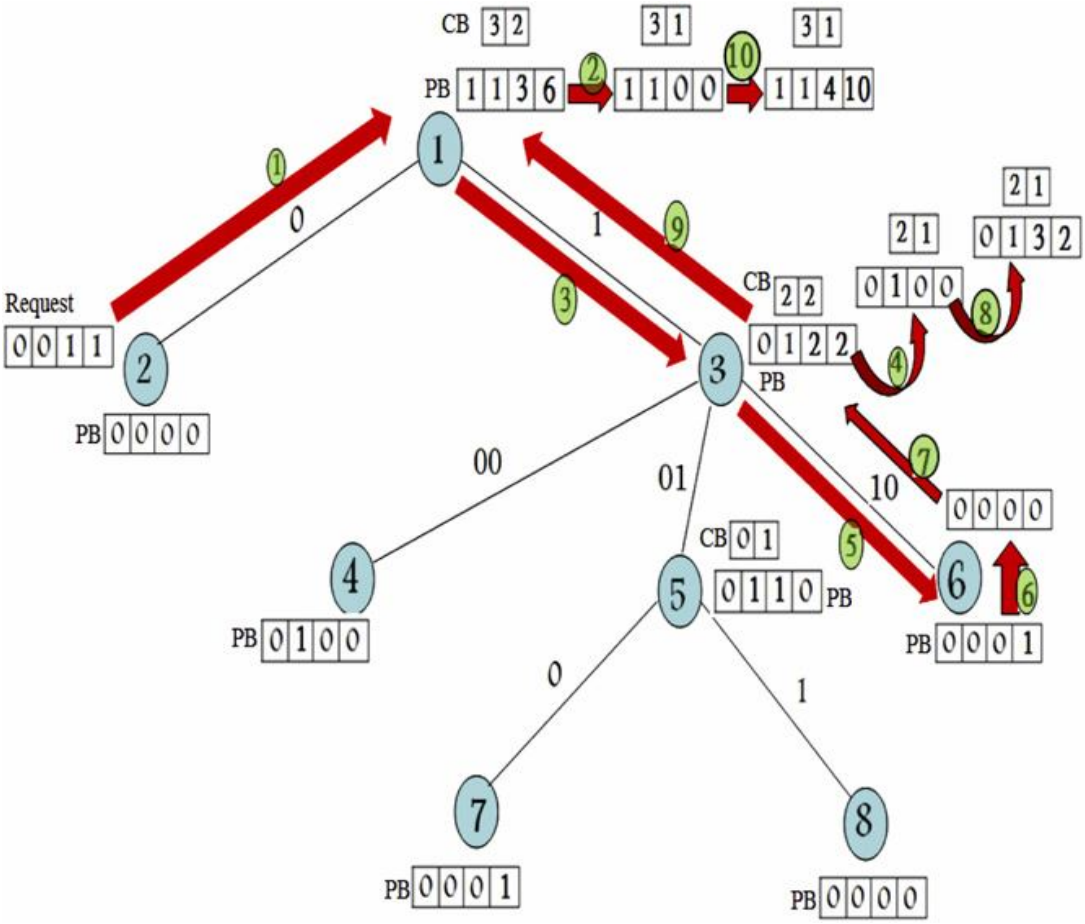
www.Prozhe.com

با اطلاعاتی مخالف ۰ رسید این اطلاعات را برداشته و از دو خانه مربوطه گره، اعداد را پاک می‌کند و به جای آن‌ها عدد ۰ قرار می‌دهد. همین عملیات در هر گره‌ای که درخواست به سمت پایین درخت حرکت می‌کند انجام می‌شود تا به گره مقصد برسد. وقتی درخواست به گره مقصد رسید اعداد ۰۱ موجود در خانه مربوطه به ۰۰ تغییر می‌یابد. علت این عمل آن است که اگر قرار است منبع مورد نظر به کاربر تحویل داده شود، نباید هیچ گره‌ای آدرس این منبع را در اختیار داشته باشد، چون اگر این منبع به کاربر تحویل داده شود، دیگر برای سایر کاربران قابل استفاده نیست.

در این روش گره‌های مسیر، مدت زمانی را بعد از اینکه درخواست کاربر تغییراتی را در نقشه‌های بیتی آن‌ها داد، منتظر می‌مانند تا یک پیغام از همان مسیری که درخواست کاربر را به آن ارسال کرده بود دریافت شود و خانه‌هایی را که در آن تغییرات ایجاد شده به روز رسانی کند (۰۰). اگر مدت زمانی را که برای این کار در نظر گرفته شده است پایان پذیرد، این گره اطلاعات قبلی خود را به محل‌های مربوطه باز می‌گرداند (Restore)، در غیر این صورت این گره، یک بسته (نقشه بیتی) دریافت می‌کند. ابتدا این گره وضعیت جفت خانه‌ای که در نقشه‌بیتی خودش تغییر کرده بود را در نقشه‌بیتی رسیده بررسی می‌کند. اگر این جفت خانه، حاوی اطلاعات یک منبع بود، محتویات این جفت خانه را در محل مربوطه در نقشه خودش قرار می‌دهد و این نقشه‌بیتی را به گره والدش ارسال می‌کند و گره والد هم به همین ترتیب تا اینکه به گره ریشه برسد، در این هنگام درخت ما به روز رسانی شده است. حالت دیگر این است که این درخت جفت خانه رسیده از فرزند محتوی آدرس هیچ منبعی نباشد (۰۰ باشد). در این حالت گره محتوای خانه مربوطه در نقشه بیتی شمارنده خود را چک می‌کند. اگر موقعیت مربوطه ۰ بود، نشان می‌دهد که این گره در فرزندان و نسل‌هایش منبع مورد نظر را در اختیار ندارد و درخواست را به سمت گره والد می‌فرستد، در غیر این صورت اگر در این موقعیت عددی غیر از صفر وجود داشته باشد نشان می‌دهد که حداقل در یکی از فرزندانش اطلاعات منبع مربوطه وجود دارد. بنابراین این گره از فرزندان خود نقشه‌های بیتی مسیرشان را می‌گیرد و آدرس منبع مورد نظر را از حداقل یکی از فرزندان یافته و آن را بعد از تغییرات لازم در جفت خانه پاک شده خود قرار می‌دهد و سپس نقشه‌بیتی مسیر را به گره والد خود تحویل می‌دهد و از گره والد به بعد نیازی نیست که اطلاعات فرزندان جمع‌آوری شود و فقط نقشه بیتی مسیری که محتوای دو خانه‌اش تغییر یافته به سمت گره ریشه می‌رود تا متوقف شود. در شکل ۳-۹ نمونه‌ای از عملیات به روز رسانی نشان داده شده است.

www.Prozhe.com

www.Prozhe.com



شکل۳-۹: نمونه‌ای از فرآیند به روز رسانی (Khanli, ۲۰۱۱).

در این شکل مراحل به ترتیب شماره‌گذاری شده است. همانطوری که در این شکل ملاحظه می- کنید درخواست کشف منبع اعداد ۳۶ را از نقشه بیتی گره ۱ پاک کرده و بجای آن عدد ۰۰ قرار می‌دهد. از محتوای خانه مربوطه در نقشه بیتی شمارنده هم یکی کم می‌کند. سپس درخواست به گره ۳ می‌رسد و در این گره نیز همین روند را تکرار می‌کند. در نهایت وقتی به گره ۶ می‌رسد، محتوای خانه مربوطه را از ۰۱ به ۰۰ تغییر می‌دهد و منبع مربوطه رزرو می‌شود. در مسیر به روز رسانی، گره ۶ نقشه بیتی تغییر یافته را به

www.Prozhe.com

www.Prozhe.com

گره ۳ ارسال می‌کند و گره ۳ با مشاهده اینکه محتوای جفت خانه مربوطه ۰۰ است، اطلاعات تمامی گره-های فرزندان خود را (به جز فرزندی که نقشه بیتی را از آن دریافت کرده) درخواست می‌کند (البته بعد از مشاهده اینکه عددی غیر از صفر در موقعیت مربوطه در نقشه‌بیتی شمارنده‌اش قرار دارد). مشخص است که اظلاعات مربوط به منبع مورد نظر را از یکی از فرزندانش دریافت خواهد کرد (گره ۵)، سپس این اطلاعات دریافتی را بعد از تغییرات لازم در خانه مربوطه قرار داده و به گره والدش یعنی گره ۱ ارسال می‌کند. گره ۱ چون مشاهده می‌کند که در خانه مربوطه در نقشه بیتی دریافتی‌اش عددی غیر از ۰۰ وجود دارد، نیازی نمی‌بیند که اطلاعات فرزندانش را جمع‌آوری کند و همان اطلاعات را بعد از تغییرات لازم در مکان‌های مربوطه ذخیره می‌کند. در این مرحله به روز رسانی روش به پایان می‌رسد و کل محیط به روز رسانی می-شود. همان‌طوری که ملاحظه شد در روش به روز رسانی ، فقط در یکی از سطوح درخت اطلاعات فرزندان جمع آوری می‌شود و در بقیه سطوح تا رسیدن به ریشه فقط یک گره ملاقات می‌شود.

۳-۶- بهبود روش در یک محیط گرید واقعی

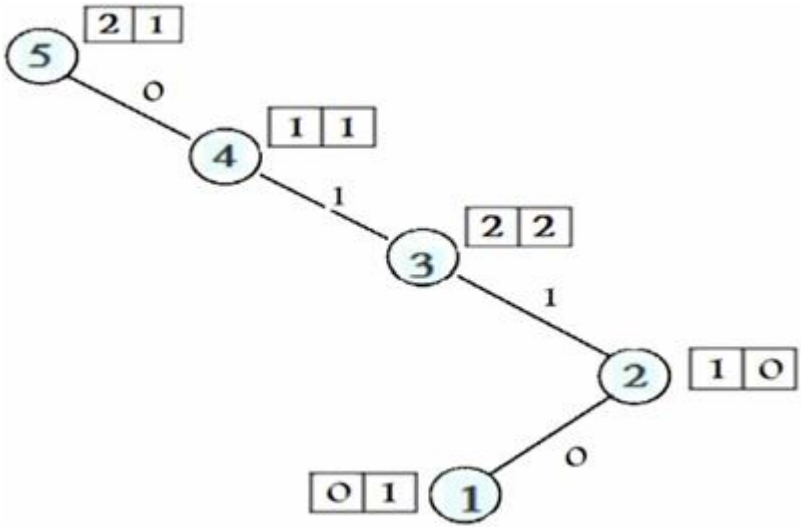
روش ما به این صورت است که در هر گره وزن یال‌ها را جمع آوری و سپس با تبدیل به عدد دهدهی معادل‌اش در مکان‌های مربوطه‌شان در گره‌ها ذخیره می‌کنیم. این روش در یک محیط گرید واقعی می‌تواند اصلاح شود زیرا اگر یک محیط گرید واقعی را با تعداد گره‌ها و لینک‌های بسیار زیاد در نظر بگیریم، تعداد صفر و یک هایی که قرار است در هر گره به هم متصل کنیم بسیار زیاد خواهد شد و در نتیجه وقتی می-خواهیم به عدد دهدهی معادلش تبدیل کنیم این یک عدد بسیار بزرگ خواهد بود که ممکن است قابل محاسبه نباشد.

ما این مشکل را به این صورت حل می‌کنیم که برای ذخیره کردن صفر و یک ها مقدار آستانه‌ای را در نظر می‌گیریم. وقتی یک منبع مطابق معمول اطلاعاتی را از گره فرزند خود می‌گیرد، این اطلاعات شامل نقشه بیتی مسیر و شمارنده است. این گره ابتدا محتوای نقشه بیتی شمارنده را بر می‌دارد و در صورت وجود منبعی در سایر فرزندانش، تعداد منابع موجود در فرزندانش را به مکان‌های مربوطه در نقشه بیتی شمارنده اضافه می‌کند. روشی را که برای پیشگیری از ذخیره کردن اعداد بسیار بزرگ در گره‌ها پیشنهاد شده به این صورت است که وقتی یک گره نقشه‌های بیتی مسیر و شمارنده را دریافت می‌کند، ابتدا منابع احتمالی

www.Prozhe.com

www.Prozhe.com

موجود در فرزندان خود را به محل‌های مربوطه در نقشه‌بیتی شمارنده اضافه می‌کند و سپس محتویات نقشه بیتی مسیر را به دودویی معادلش تبدیل کرده و وزن یال را به آن اضافه می‌کند. اگر این عدد دودویی از مقدار آستانه‌ای که در نظر گرفته شده بود بیشتر شد، این گره به جای ذخیره کردن این عدد بزرگ فرض می‌کند گره‌ای که این عدد بزرگ را از آنجا دریافت کرده یک گره مقصد باشد (یعنی در این گره وجود داشته باشد)، بنابراین تنها وزن یال مربوطه را به همراه شمارنده ذخیره می‌کند و در واقع شمارنده موجود در نقشه بیتی مسیر از اول شروع می‌شود و این اطلاعات به گره‌های والد فرستاده می‌شود و به همین ترتیب. حال وقتی درخواست به یکی از این گره‌ها رسید و به سمت گره مقصد راهنمایی شد باید در گره مقصد مقایسه‌ای انجام دهد که آیا عدد ذخیره شده در گره مقصد ۰۱ است یا خیر. اگر ۰۱ باشد یعنی منبع در همین گره موجود دارد وگرنه عددی که در این گره وجود دارد را برداشته و به سمت گره مقصد حرکت می‌کند و به همین ترتیب. شکل ۳-۱۰ همین روند را نشان می‌دهد. در اینجا مقدار آستانه برابر با ۲ در نظر گرفته شده است.



شکل۳-۱۰: نمونه‌ای از یک درخت کشف منبع ردپا با مقدار آستانه ۲(Khanli, ۲۰۱۱).

www.Prozhe.com

www.Prozhe.com

۳-۷- پیچیدگی مکانی و زمانی

همانطوری که می‌دانیم یک مقیاس خوب برای سنجش کارایی یک الگوریتم پیچیدگی زمانی و مکانی آن الگوریتم است. در این بخش ما پیچیدگی الگوریتم خودمان را از لحاظ مکانی و زمانی بررسی می‌کنیم. فرض کنید که n گره در محیط وجود داشته باشد و r نوع منبع در محیط باشد. هر منبع a صفت داشته باشد و طول نقشه های بیتی b بیت باشد. همچنین با فرض اینکه حداکثر تعداد فرزندان هر نود در محیط m باشد، پیچیدگی الگوریتم ما به صورت زیر خواهد بود.

پیچیدگی زمانی را می‌توان به صورت زمان مصرفی در بین گره‌ها و زمان‌های مصرفی در خود گره‌ها در نظر گرفت. در اینجا ما فرض می‌کنیم که زمان مصرفی در بین گره‌ها بیشتر باشد. بنابراین با محاسبه تعداد لینک‌های اشغال شده می‌توان به نتایجی در مورد زمان مصرفی رسید. در الگوریتم ما در بدترین حالت در کشف منبع از گره برگ تا گره ریشه در هر سطح فقط یک گره و از گره ریشه تا به گره برگ در هر سطح یک گره ملاقات می‌شوند. بنابراین در بدترین حالت پیچیدگی زمانی الگوریتم ما $O(\log_m n)$ خواهد بود که در مقایسه با روش‌های دیگر همچون (Chang, ۲۰۱۰) بسیار کمتر است. این نشان می‌دهد که در روش ما منبع درخواستی کاربر در زمان کوتاه‌تری برای کاربر کشف می‌شود.

پیچیدگی مکانی الگوریتم ما هم مکانی است که در بدترین حالت توسط نقشه‌های بیتی که توسط روش ما استفاده می‌شود در حافظه گره‌ها اشغال می‌شود. این مقدار برای الگوریتم ما O(nrab) خواهد بود.

۳-۸- خلاصه و نتیجه‌گیری

در این فصل ما یک روش کشف منبع در محیط گرید ارائه دادیم که موضوع اصلی پایاننامه ما نیز می‌باشد. در این روش ما برای اولین بار از یک درخت وزن‌دار برای کشف منبع در محیط گرید استفاده کردیم. ما همچنین روش کشف منبعی را ارائه دادیم که با استفاده از وزن‌های درخت قادر است مسیر رسیدن به هدف را به طور مستقیم و بدون مراجعه به گره‌های غیرضروری و اضافی کشف کند. هزینه به روز رسانی روش پیشنهادی نیز پایین است. در این فصل همچنین روش‌هایی را نیز برای بهبود روش پیشنهادی خودمان نیز ارائه داده‌ایم تا روش پیشنهادی‌مان هرچه بیشتر قابل استفاده در یک محیط گرید واقعی باشد. روشی را که

www.Prozhe.com

www.Prozhe.com

ما ارائه داده‌ایم نسبت به روش‌های قبلی کشف منبع را تا حد زیادی بهبود می‌دهد. نتایج در بخش بعد آورده شده است و کارایی الگوریتم ما را در مقایسه با روش‌های دیگر به خوبی نشان می‌دهد.

به طور کلی در این کار هدف ما پایین آوردن هزینه کشف منبع در یک محیط گرید بود. دلیل این امر هم اندازه بسیار بزرگ محیط گرید نسبت به سایر محیط‌ها و همچنین وجود کاربران و درنتیجه درخواست‌های فراوان می باشد. الگوریتم‌های کشف منبعی می‌توانند برای چنین محیط‌هایی قابل استفاده باشند که کشف منبع را با ملاقات گره‌های بسیار کم و تولید ترافیک کم انجام داده و در نتیجه در یک زمان بسیار کم منبع مورد نیاز کاربر را کشف کرده و رضایت کاربران را جلب کنند. در فصل بعدی نتایجی را که در شبیه‌سازی‌ها بدست آمده است را در نمودارهایی آورده‌ایم. با این نمودارها ما می‌توانیم کارایی الگوریتم پیشنهادی خودمان را با سایر روش‌ها مقایسه کنیم.

www.Prozhe.com

www.Prozhe.com

فصل چهارم

www.Prozhe.com

www.Prozhe.com

نتایج شبیه سازی

۴-۱- مقدمه

در این فصل برای اینکه کارایی روش خود را بهتر نشان دهیم یک سری شبیه‌سازی‌هایی را انجام داده‌ایم و روش پیشنهادی خودمان را از لحاظ معیارهای مختلفی مورد ارزیابی قرار داده‌ایم و نتایجی را بدست آورده-ایم. ما این نتایج را در قالب نمودارهایی نشان داده‌ایم و این نتایج را با روش‌های مختلفی مقایسه کرده‌ایم. همانطوری که ملاحظه خواهیم کرد نتایج کارایی روش ما را نسبت به روش‌های دیگر به خوبی نشان می‌دهد.

در شبیه‌سازی‌ها ما روش خودمان را با روش پیشنهاد شده در مرجع (Chang, ۲۰۱۰)، که ما آن را روش درختی (Tree method) نامگذاری کرده‌ایم، همچنین با روش Flooding-based و الگوریتم پیشنهاد شده در مراجع (Marzolla, ۲۰۰۵; Marzolla, ۲۰۰۷) یعنی MMO مقایسه کرده‌ایم.

در شبیه‌سازی‌ها ما همانند مراجع (Mastroianni, ۲۰۰۷) و (Chang, ۲۰۱۰) ارتفاع درخت‌ها را ۳ و ۴ در نظر گرفتیم و شبیه‌سازی‌ها را بر روی درخت‌هایی با ارتفاع ۳ و ۴ انجام دادیم. در آزمایشات ما تعداد گره‌های موجود در محیط را متغیر در نظر گرفته و همچنین تعداد درخواست‌های مشخص را به صورت تصادفی به گره‌های موجود در محیط تحویل می‌دهیم. ما در شبیه‌سازی‌ها تعداد گره‌های ملاقات شده در روش‌های مختلف را به ازای درخواست‌های مشخص محاسبه کرده و در نمودارها نشان داده‌ایم. همانطوری که مشخص است هر چقدر در یک روش تعداد گره‌های ملاقات شده در مراحل کشف منبع یا به روزرسانی کمتر باشد آن روش موثرتر است. همچنین ما آزمایشی را نیز برای سنجش میزان ترافیکی را که روش‌ها در محیط ایجاد می‌کنند، انجام داده‌ایم. در این شبیه‌سازی ترافیکی را که روش‌های مختلف ایجاد می‌کنند را محاسبه و با روش خودمان مقایسه کرده‌ایم. در این آزمایش نیز هر روش مقادیر کمتری را در نمودارها نشان دهد یعنی ترافیک کمتری را در محیط ایجاد می‌کند و در نتیجه روش موثرتری است. در بخش بعدی ما نتایج مربوط به شبیه‌سازی‌ها را آورده‌ایم.

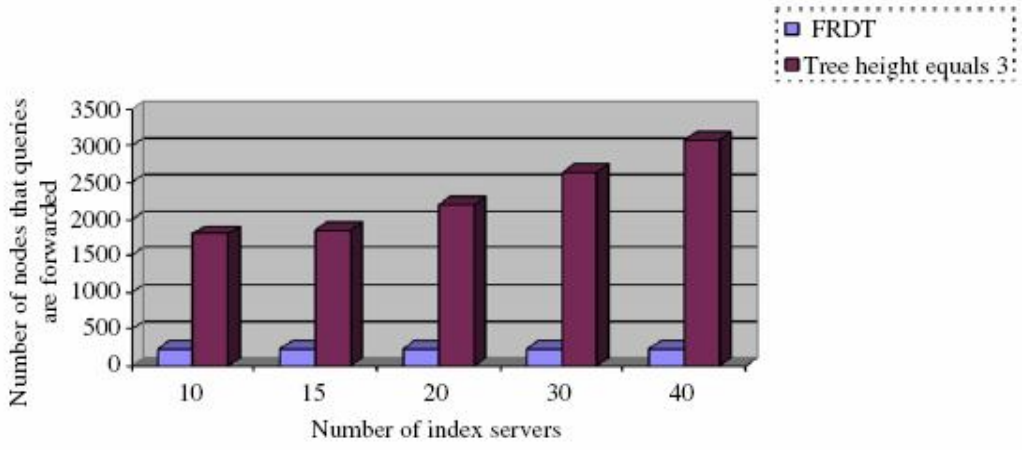
www.Prozhe.com

www.Prozhe.com

۴-۲- نتایج شبیه سازی

ما شبیه‌سازی‌ها را در محیط MATLAB انجام داده‌ایم. همانطوری که در بخش قبل نیز بحث شد ارتفاع درخت‌ها را در تمام شبیه‌سازی‌ها ۳ یا ۴ درنظر گرفته‌ایم.

در اولین آزمایش، ما روش پیشنهادی خودمان(FRDT) را با روش درختی (Tree method) به ازای تعداد index server های مختلف مقایسه کرده‌ایم. در این آزمایش تعداد ۲۰۰ گره در محیط در نظر گرفته‌ایم و تعداد درخواست‌هایی را که در محیط تولید خواهد شد را ۱۰۰ درنظر گرفتیم. ما منابع را به صورت تصادفی بین گره‌های موجود در محیط پخش می‌کنیم و سپس درخواست‌ها را به ترتیب و به صورت تصادفی به هر یک از گره‌های موجود در محیط ارسال می‌کنیم. ما تعداد گره‌هایی را که در هر روش برای کشف منبع در محیط ملاقات می‌شوند را محاسبه و در قالب نمودار نشان داده‌ایم. در شکل ۴-۱ تفاوتی را که در تعداد گره-های ملاقات شده در روش پیشنهادی ما نسبت به روش مقایسه شده یعنی درختی وجود دارد را ملاحظه می‌کنید. این یک تفاوت بسیار زیاد است که با افزایش تعداد index server ها این اختلاف بیشتر می‌شود، در حالی که در روش ما این یک مقدار ثابتی است. این به دلیل آن است که روش ما مسیر مستقیمی را تا رسیدن به هدف طی می‌کند. نتایج این شبیه‌سازی در شکل ۴-۱ نشان داده شده است.

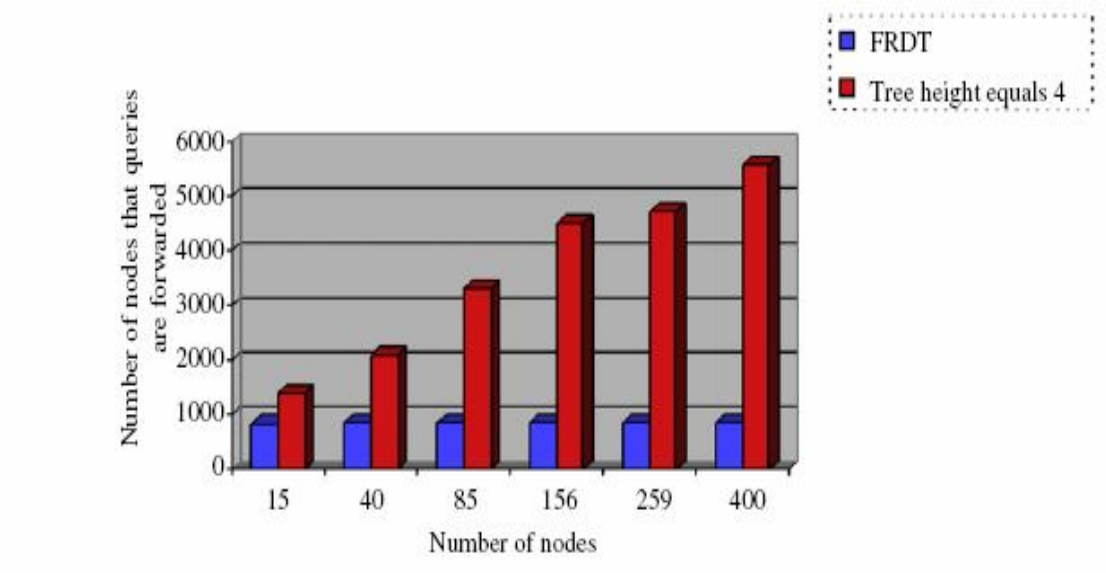


شکل۴-۱: تعداد گره‌هایی که توسط درخواست‌های کشف منبع ملاقات می‌شوند (Khanli, ۲۰۱۱).

www.Prozhe.com

www.Prozhe.com

در شبیه سازی دوم ما فرض کرده‌ایم که ۳۰۰ درخواست در محیط توسط کاربران تولید شود و همچنین در این شبیه‌سازی ارتفاع درخت را ۴ در نظر گرفتیم. در شکل ۴-۲، مقایسه‌ای را بین دو روش FRDT و Tree methodبه لحاظ تعداد گره‌های ملاقات شده در مرحله کشف منبع توسط درخواست‌های کشف منبع ملاحظه می‌کنید. در این شبیه‌سازی ما تعداد گره‌های موجود در محیط را متغیر درنظر گرفته‌ایم تا کارایی روش‌ها در محیط‌هایی با اندازه متفاوت سنجیده شود. همانگونه که ملاحظه می‌کنید در این شبیه سازی‌ها نیز روش ما حتی با افزایش تعداد گره‌های محیط، تعداد گره‌های ثابتی را ملاقات می‌کند. در این آزمایش نیز تفاوت روش ما نسبت به روش مورد مقایسه کاملا مشخص است و روش پیشنهادی ما گره های بسیار کمتری را نسبت به روش دیگر ملاقات می‌کند.



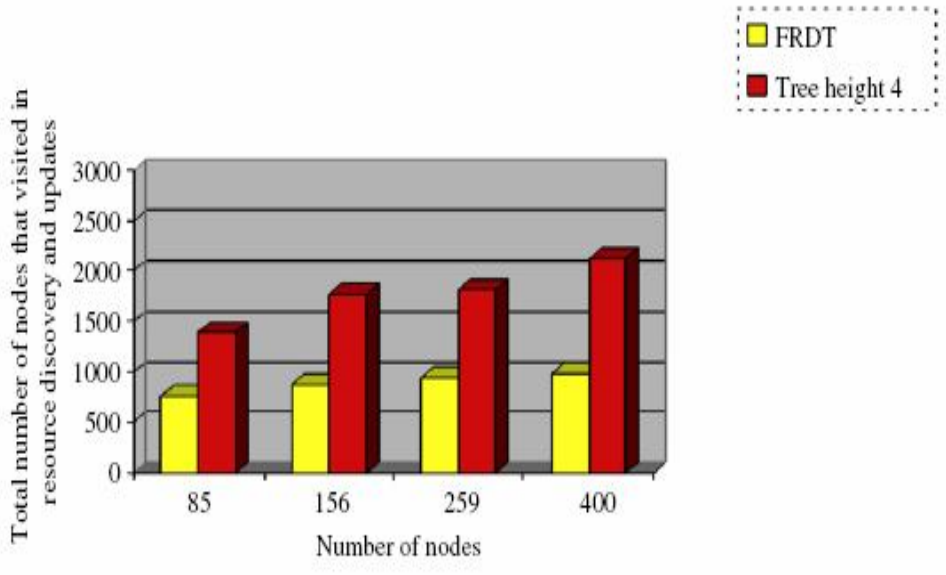
شکل۴-۲: تعداد گره‌هایی درخواست‌های کاربران به آن‌ها فرستاده می‌شوند (Khanli, ۲۰۱۱).

در شبیه‌سازی سوم، ما این بار مجموع تعداد گره‌های ملاقات شده در دو روش FRDT وTree method را در طول مراحل کشف منبع و به روز رسانی محاسبه کرده‌ایم. در این شبیه‌سازی ابتدا ۱۰۰ درخواست را در محیط پخش می‌کنیم و هر درخواست در هر گره‌ای که منبعی را کشف می‌کند و منبع به کاربر فرضی

www.Prozhe.com

www.Prozhe.com

اختصاص داده می‌شود بلافاصله اطلاعات این منبع را در درخت به روز رسانی می‌کنیم. ما این کار را برای هر ۱۰۰ درخواست انجام داده‌ایم. ارتفاع درخت در این شبیه‌سازی ۴ درنظر گرفته شده است. همانگونه که در شکل ۴-۳ مشاهده می‌کنید روش پیشنهادی ما باز تعداد گره‌های کمتری را نسبت به روش درختی ملاقات می‌کند و این اختلاف با افزایش تعداد گره‌های محیط بیشتر می‌شود. این یعنی روش ما در محیط-هایی با گره‌های زیاد نسبت به روش درختی موثرتر است.



شکل۴-۳: مجموع تعداد گره‌های ملاقات شده در مراحل کشف منبع و به روز رسانی (Khanli, ۲۰۱۱).

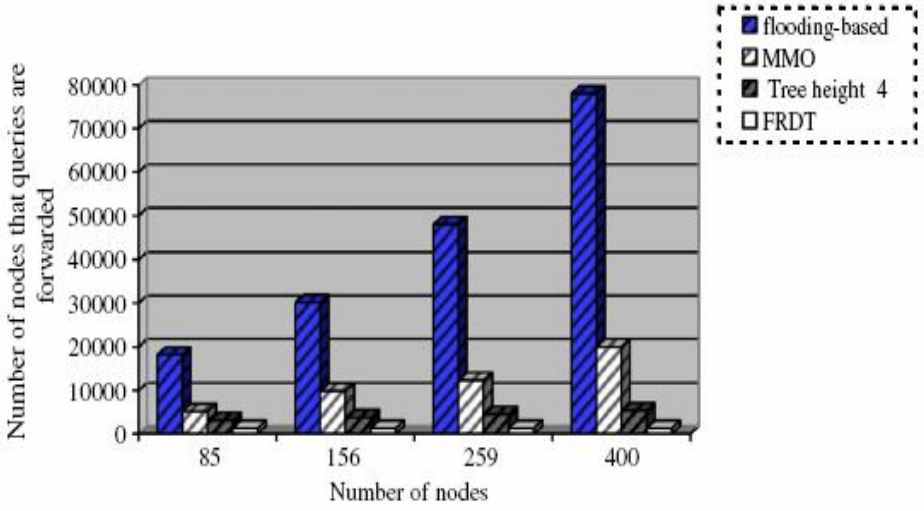
در شبیه‌سازی بعدی، متوسط تعداد گره‌های ملاقات شده را در کشف منبع این بار برای روش‌های بیشتری محاسبه کرده و به صورت نمودار آورده‌ایم. در این شبیه‌سازی روش‌های Flooding-based ، MMO، و Tree methodروش پیشنهادی خودمان یعنی FRDT مورد ارزیابی قرار گرفته‌اند و نتایج در شکل ۴-۴ آورده شده است. همانگونه که ملاحظه می‌کنید روش Flooding-based بیشترین تعداد گره‌های ملاقات شده را دارد، چون در این روش درخواست‌ها به صورت سیل‌آسا به تمام گره‌ها فرستاده می‌شوند. روش MMO به خاطر اینکه اطلاعات بیشتری را نسبت به روش Flooding-based نگه می‌دارد گره‌های کمتری را ملاقات می‌کند. در

www.Prozhe.com

www.Prozhe.com

نهایت Tree method توانسته از تمام روش‌های قبلی خود نتایج بهتری داشته باشد، ولی روش FDRT حتی نسبت به Tree method تعداد گره‌های بسیار کمتری را ملاقات می‌کند. از این شبیه‌سازی‌ها که در محیط-هایی با اندازه‌های مختلف انجام گرفته است می‌توان دید که روش FRDT نسبت به سایر روش‌ها موثرتر است.

در آزمایش پنجم تعداد لینک‌هایی را که در روش FRDT در طول مراحل کشف منبع و به روز رسانی اشغال می‌شود را برای ۳۰۰ و ۱۰۰۰ درخواست ملاحظه می‌کنید. در شکل ۴-۵ تعداد لینک‌های اشغال‌شده برای ۳۰۰ درخواست در محیط با افزایش تعداد گره‌های محیط نشان داده شده است و در شکل

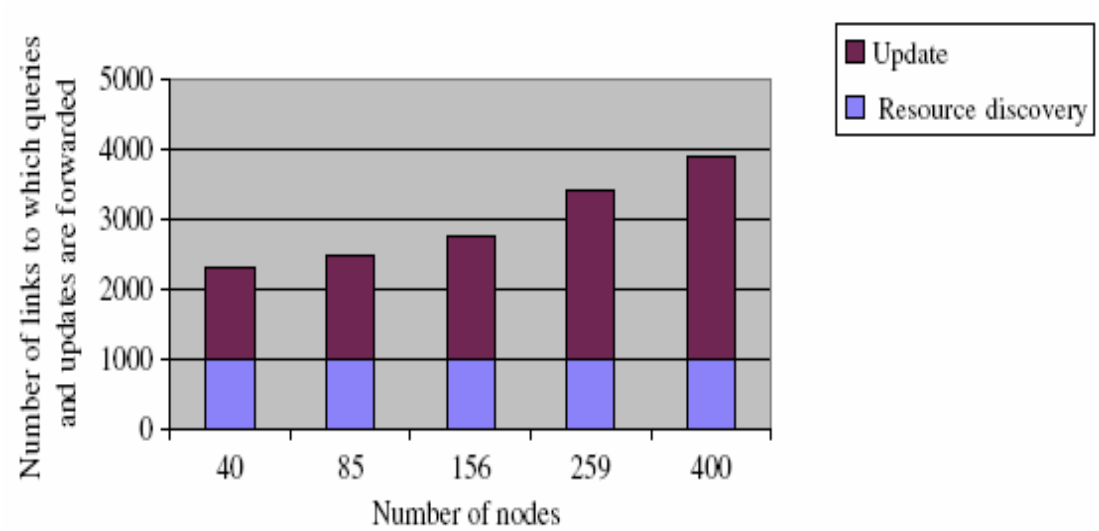


شکل۴-۴: متوسط تعداد گره‌هایی که توسط درخواست‌های کشف منبع برای روش‌های مختلف ملاقات می‌شوند (Khanli, ۲۰۱۱).

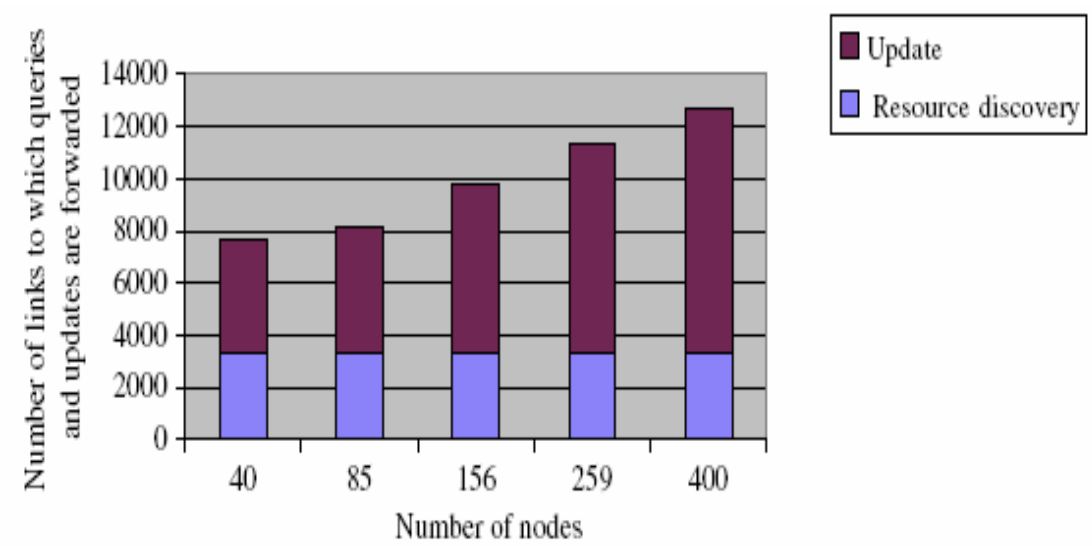
۴-۶ همان آزمایش قبلی این بار با تعداد ۱۰۰۰ درخواست در محیط نشان داده شده است. این آزمایش جزیئات بیشتری را برای روش پیشنهادی ما نشان می‌دهد.

www.Prozhe.com

www.Prozhe.com



شکل۴-۵: تعداد لینک‌های ملاقات شده توسط درخواست‌های کشف منبع و به روز رسانی به ازای ۳۰۰ درخواست در محیط (Khanli, ۲۰۱۱).



شکل۴-۶: تعداد لینک‌های ملاقات شده توسط درخواست‌های کشف منبع و به روز رسانی به ازای ۱۰۰۰ درخواست در محیط (Khanli, ۲۰۱۱).

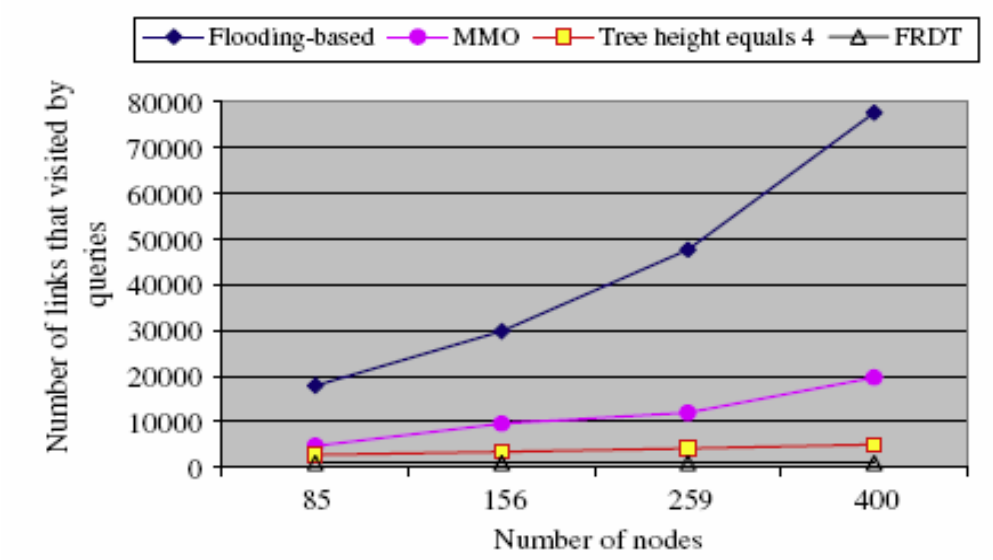
www.Prozhe.com

www.Prozhe.com

در آزمایش آخر، ما ترافیک تولید شده در محیط گرید را در مرحله کشف بـرای روش هـای Flooding-based ، MMO، Tree method و FRDT محاسبه و به صورت نمودار در شکل ۴-۷ نشان داده‌ایم. در این شبیه‌سـازی-ها تعداد گره‌های موجود در محیط را متغیر در نظر گرفتیم و ۳۰۰ درخواست را در محیط تولید و بین گره‌ها پخش کرده‌ایم. در این شکل هر روشی که مقدار کمتری را نشان دهد یعنـی آن روش ترافیـک کمتـری را در محیط تولید می‌کند و بنابراین روش موثرتری است. همانگونه که ملاحظه می‌کنید در این آزمایش نیـز روش FRDT نسبت به بقیه روش‌ها ترافیک کمتری را تولید می‌کند و بنابراین روش موثرتری است.

۴-۳- نتیجه گیری

همانگونه که در بخش نتایج ملاحظه کردید ما کـارایی روش خودمـان را بـا انجـام دادن آزمایـشات مختلفـی سنجیده ایم و برای اینکه این کارایی بهتر مشخص شود این نتایج را با نتایج بدست آمده از روش



شکل۴-۷: تعداد لینک های ملاقات شده توسط درخواست های کشف منبع به ازای ۳۰۰ درخواست توسط روش های مختلف (Khanli, ۲۰۱۱).

www.Prozhe.com

www.Prozhe.com

های مختلف نیز مقایسه کرده‌ایم. نتایج مقایسه نشان دادند که روش پیشنهادی ما چه در تعداد گره‌های ملاقات شده در مراحل کشف منبع و به روز رسانی، چه در تعداد لینک‌های اشغال شده و همچنین ترافیک تولیدی در محیط نسبت به بقیه روش ها موثرتر و بهتر است. ما در این شبیه‌سازی‌ها تعداد گره‌های موجود در محیط را افزایش دادیم تا کارایی روش‌ها در محیط‌هایی با گره‌های زیاد نیز سنجیده شود و همانگونه که ملاحظه کردید روش ما در محیط‌هایی با اندازه بزرگتر نیز کارایی خود را حفظ می‌کند و از بقیه روش‌ها موثرتر است. هدف ما از این شبیه‌سازی‌ها نشان دادن معیارهایی بود که نشان دهد روش ما می‌تواند در محیط‌هایی با اندازه متفاوت نسبت به برخی از روش‌هایی که اخیرا ارائه شده‌اند روش موثرتری است. در فصل بعد ما خلاصه‌ای از آنچه را که در این پایاننامه بحث شد را خواهیم آورد و همچنین برخی از کارهایی را که می‌توان در جهت بهبود روش خودمان انجام داد تا به روش کامل‌تری دست یافت را پیشنهاد داده‌ایم.

www.Prozhe.com

www.Prozhe.com

فصل پنجم

www.Prozhe.com

www.Prozhe.com

vhdl

۵-۱- مقدمه

زبان توصیف سخت افزار می تواند فعالیت عمده خود را روی وسایل منطقی قابل برنامه ریزی (programming logic devices) یا پیچیده تر(cplds) و (field programmable gate arrays) داشته باشد. چند گونه از این زبان وجود دارد مانند vhdl ، verilog ، abel که ما در این بخش آموزشی به vhdl خواهیم پرداخت. نرم افزار های بسیاری برای طراحی vhdl ساخته شده که یکی از آنها active-vhdl از سوی شرکت aldec می باشد که یادگیری آن بسیار ساده است. گر چه vhdl توسط وزارت دفاع آمریکا به منظور اهداف نظامی تولید شد ولی به سرعت گسترش یافت و با یک استاندارد خاص (the institute of electrical and electronic engineers : IEEE) مورد استفاده عموم قرار گرفت که IEEE به مرور از سال ۱۹۸۷ تا ۱۹۹۸ کامل شد. زبان vhdl ابتدا به منظور شبیه سازی و مدل سازی و درک بیشتر مدارهای منطقی بوده است که توسط محققان عمل synthesis یا سنتز به عنوان اتوماتیک کردن فرآیند طراحی به آن اضافه شده است.

۵-۲- فاکتورهای قابل ملاحظه در ارزش vhdl برای یک طرح

۱- محبوبیت جهانی زبان hdl (زبان توصیف سخت افزار)

زبان vhdl در شمال آمریکا و همچنین اروپا توسط ۸۰ درصد مهندسین سیستم استفاده می شود که این رقم همچنان در حال رشد است.

۲- وجود انواع مختلف روشهای توصیف در این زبان

Vhdl برای کاربر از نظر یکتا بودن نوع توصیف هیچ گونه محدودیتی ندارد در واقع یک برنامه را می توان هم به صورت رفتاری و هم به صورت متنی یعنی ذکر تمام گیتهای موجود به کار برد همچنین از vhdl می توان در سطوح مختلفی از پیچیدگی استفاده کرد از یک ترانزیستور کوچک گرفته تا یک سیستم کامل را می توان با آن طراحی کرد.

۳- وجود نرم افزارهای شبیه ساز vhdl

www.Prozhe.com

www.Prozhe.com

یکی از دلایل افزایش محبوبیت این زبان وجود تعدادی از نرم افزار های شبیه ساز با قیمتی مناسب مانند active vhdl است که می توانید با مراجعه به سایت www.aldec.com در مورد آن بیشتر اطلاعات کسب کنید.

۴- در اختیار داشتن سنتزکننده های vhdl

امروزه بسیاری از فروشندگان وسایل قابل برنامه ریزی (programmable device) و شرکت های تولید کننده نرم افزارهای EDA سنتزکننده های گوناگون را ارائه داده اند مانند active-cad که یک سنتز کننده منطقی برای تمام مدارات مجتمع FPGA و CPLD می باشد.

۵- vhdl یک زبان مدلسازی جهانی

گرچه vhdl بیشتر ما را متوجه الکترونیک می کند در واقع به یک زبان مدل سازی جهانی شهرت یافته است و برای مدل سازی و شبیه سازی سیستمهای الکترومکانیکی و هیدرولیکی و شیمیایی و ...مورد استفاده قرار می گیرد.

چند نکته مهم در مورد کد نویسی در vhdl

۱- زبان vhdl یک زبان حساس به حروف نیست یعنی حروف بزرگ و کوچک را بدون تفاوت در نظر می گیرد مثلا کلمات زیر هیچ فرقی با هم ندارند: architecture ، ARCHITECTURE ، Architecture

۲- نکته شروع ترجمه ی کدهای شما در اول هر خط از دستورات می باشد یا به عبارتی دیگر اجرای دستورات در vhdl مانند cpp ترتیبی نیست یعنی دستورات یکباره و با هم اجرا می شوند.

اکنون به برنامه زیر که به زبان vhdl نوشته شده توجه کنید شما باید کدهای vhdl را دقیقاً دنبال کنید تا یک دید کلی در مورد روش و آن بدست آورید.

```
Library IEEE;
Use IEEE.std_logic_۱۱۶۴.all;
Entity prime is
Port (N۰, N۱, N۲, N۳: in BIT ;
F: out BIT);
End prime;
Architecture prime۱_arch of prime is
Signal and۱, and۲, and۳, and۴: bit;
```


www.Prozhe.com

www.Prozhe.com

```
Begin
and۱<= (not n۳) and n۰;
and۲<= (not n۳) and (not n۲) and n۱;
and۳<= (not n۲) and n۱ and n۰;
and۴<= n۲ and (not n۱) and n۰;
f <= and۱ or and۲ or and۳ or and۴;
End prime۱_arch;
```

۵-۳- تعریف ورودی و خروجی

۱- Entity : اکنون وقت آن رسیده که با اولین دستور آشنا شوید : entity

همیشه هر سیستمی در طراحی vhdl با Entity آغاز خواهد شد .

مرکز اصلی برنامه شما یا به عبارت دیگر همان main شما بین دو عبارت Entity ، end Entity; تعریف می شود.نام سیستمی که شما قصد طراحی آن را دارید مقابل کلمه ی Entity قرار می گیرد مثلاً اگر می خواهید یک processor طراحی کنید باید بنویسید :

```
Entity processor is
این قسمت محل تعریف ورودی ها و خروجی های برنامه شماست- -
End entity;
```

حالا باید ببینیم منظور از ورودی و خروجی که بعد از entity تعریف می شود چیست ؟

۱- پارامترها : مانند پهنای یک گذرگاه یا bus برای یک پردازنده یا ماکزیمم مقدار فرکانس برای clock سیستم

۲- راه های ارتباطی از خارج به سیستم و از داخل سیستم به خارج که وظیفه انتقال داده ها را بر عهده دارند.

۲- Architecture :

مثال:وسایل پیشرفته الکترونیکی مانند تلویزیون گاهی کمتر استفاده می شود زیرا اگر وسیله ای برای کنترل کردن و ارتباط برقرار کردن با آنها در اختیار نداشته باشیم.این وسیله قسمت مهمی در یک طراحی vhdl می باشد مثلا در tv همان remote control را فرض می کنیم architecture همان دقیقاً کار آن را انجام می دهد که به صورت زیر تعریف می شود :

```
Entity tvset is
```


www.Prozhe.com

www.Prozhe.com

```
…
End entity tvset;
Architecture tv۲۰۰۰ of tvset is
…
End Architecture tv۲۰۰۰;
```

دستوراتی که ما می توانیم در قسمت Architecture قرار دهیم دو نوع هستند :

۱- functionality (behavioral)

۲- structural

منظور از رفتاری (behavioral) این است که دستورات ما فقط رفتار سیستم را توصیف کنند و به انواع گیتهای به کار گرفته شده و ریزه کاری ها در سیستم کاری نداریم. سنتز کردن اینگونه برنامه ها توسط نرم افزار های سنتز کننده دشوار خواهد بود ولی درک و فهم برنامه ای که ما به این روش نوشته ایم بسیار آسان است.

منظور از ساختمانی (structural) یعنی دستورات ما از نظر مدار های منطقی کامل باشند و تمام گیتهای موجود در سیستم را توصیف می کنیم که به این برنامه نویسی کامل گفته می شود.سنتز کردن اینگونه برنامه ها آسان می باشد ولی درک آنها توسط ما کاری دشوار خواهد بود.

هر سیستمی که ما قصد طراحی آن را داریم فقط یک Entity دارد با چند Architecture .

عکس این مطلب درست نیست یعنی برای یک architecture نمی توانیم تعدادی Entity داشته باشیم.

به مثال زیر توجه کنید :

اگر مینبوردی (mainboard) که با پردازنده های مختلف کار میکند را به عنوان یک برنامه vhdl در نظر بگیرید خواهید دید که مینبورد ما همان قسمت Entity برنامه است چون فقط یک مینبورد داریم پس فقط یک Entity هم داریم و پردازنده های مختلف همان Architecture های برنامه ما می باشند مثلا برای CPU های Cyrix ، Intel ، Amd خواهیم داشت :

```
Entity Pentium is
…
End entity Pentium;
Architecture Amd of Pentium is
..
End Amd;
```


www.Prozhe.com

www.Prozhe.com

```
Architecture Intel of Pentium is
..
End Intel;
Architecture Cyrix of Pentium is
..
End Cyrix;
```

با داشتن یک چنین برنامه ای هر زمان به هر نوع پردازنده ای که احتیاج داشتیم می توانیم آن را روی مینبورد وصل کنیم. همه دستوراتی که می توان در Architecture استفاده کرد در کتابخانه ای به نام IEEE وجود دارد با مراجعه به آن میتوان ابهامات احتمالی را رفع کرد.اگر به لفظی غیر استاندارد و نا مفهوم بر خوردید ابتدا LIBRARY را صدا زده سپس در کتابهای موجود در آن به دنبال کلمه مورد نظر بگردید. بسته بندی ها یا Packages که در Library وجود دارند به سه دسته تقسیم می شوند :

۱- standard

این کتابخانه به صورت Default در نظر گرفته می شود و اگر در ابتدای برنامه نام کتابخانه مورد نظر ذکر نشد کامپایلر به طور پیش فرض آن را standard در نظر می گیرد که شامل تمام انواع استاندارد عملگرها و اشیا می باشد.

۲- Textio

این library فقط به منظور شبیه سازی و مدل سازی با vhdl به کار می رود و به عنوان برنامه ای که بتواند روی مدارات مجتمع پیاده سازی شود نیست و بدین صورت در ابتدای برنامه نوشته می شود :

```
Library Std;
Use std.textio.all;
```

۳- STD_LOGIC_۱۱۶۴

این مهمترین و پرکاربردترین کتابخانه است و همیشه به عنوان مرجع استفاده می شود نمایش آن به این صورت است :

```
Library IEEE;
Use IEEE.Std_Logic_۱۱۶۴.all;
```

۵-۴- تحلیل یک برنامه vhdl

برنامه یک full adder چهاربیتی :

```
Library IEEE
```


www.Prozhe.com

www.Prozhe.com

```
Use IEEE.STD_LOGIC_۱۱۶۴.all;
Entity Adder is
Port(
I۱ : in STD_LOGIC_VECTOR(۳ downto ۰);
I۲ : in STD_LOGIC_VECTOR(۳ downto ۰);
Carry : out STD_LOGIC;
Sum : out STD_LOGIC_VECTOR(۳ downto ۰)
);
End adder;
Architecture fulladder of adder is
Signal c۱,c۲,c۳:std_logic ;
Begin
C۱<=i۱(۰) and i۲(۰);
Sum(۰)<=i۱(۰) xor i۲(۰);
C۲<=(i۱(۱)and i۲(۱)) or (i۱(۱)and c۱) or (i۲(۱)and c۱);
Sum(۱)<=i۱(۱) xor i۲(۱) xor c۱ ;
C۳<=(i۱(۲)and i۲(۲)) or (i۱(۲)and c۲) or (i۲(۲)and c۲);
Sum(۲)<=i۱(۲) xor i۲(۲) xor c۲;
Carry<=(i۱(۳)and i۲(۳)) or (i۱(۳)and c۳) or (i۲(۳)and c۳);
Sum(۳)<=i۱(۳) xor i۲(۳) xor c۳ ;
End fulladder ;
```

۵-۵- انواع داده ها data types

الف- Boolean

حتماً با این نوع داده آشنا هستید : درست و غلط یا به عبارتی false و truee نحوه اعلان آن به صورت زیر است :

```
Type Boolean is ( false , true );
```

ب- character

نحوه اعلان :

```
Type character is ( null , soh , … , 'a','b',… );
```

ج- integer

www.Prozhe.com

www.Prozhe.com

نحوه اعلان :

Type integer is range -۲۱۴۷۴۸۳۶۴۷ to ۲۱۴۷۴۸۳۶۴۷;

د- real

نحوه اعلان :

Type real is range -۱,۰e۳۰۸ to ۱,۰e۳۰۸;

ه- bit

نحوه اعلان :

Type bit is ('۰' , '۱');

۵-۶- تعریف فیزیکی یک نوع داده دلخواه

کمیتهای فیزیکی مانند زمان (time)، فاصله (distance) ، جریان (current) ، دما (temperature) و غیره را می توان تعریف کرد به مثال زیر که در مورد تعریف time می باشد توجه کنید :

```
Type time is range -۲۱٤۷٤۸۳٦٤۷ to ۲۱٤۷٤۸۳٦٤۷
   Units
        Fs:
        Ps = ۱۰۰۰ fs;
        Ns = ۱۰۰۰ ps;
        Us = ۱۰۰۰ ns;
        Ms = ۱۰۰۰ us;
        Sec = ۱۰۰۰ ms;
        Min = ٦۰ sec;
        Hr = ٦۰ min;
   End units;
```

مثال : می خواهیم برنامه ای بنویسیم که سه بیت x,y,z را به عنوان ورودی دریافت کرده و حاصل جمع آنها را در خروجی s و carry آن را در خروجی c قرار می دهد .

```
Entity Sum is
  Port ( x , y , z : in Bit ;
         S , C : out Bit );
End entity;
Architecture RTL of Sum is
  Signal Sig : Bit ;
   Begin
          Sig<=x xor y ;
```


www.Prozhe.com

www.Prozhe.com

```
        S <=Sig xor z ;
        C <=(x and y) or (y and z) or (x and z) ;
End RTL;
```


www.Prozhe.com

www.Prozhe.com

فصل ششم

www.Prozhe.com

www.Prozhe.com

نتیجه گیری و کارهای آینده

۶-۱- نتیجه گیری

همانگونه که در این پایاننامه ملاحظه کردید ما به یکی از موضوعات بسیار مهم و حیاتی در سیستم‌های گرید که امروزه تحقیقات بسیاری از محققین زمینه گرید را به خود اختصاص داده است یعنی مسئله کشف منبع در محیط گرید پرداخته‌ایم. در این کار ما برای کشف منبع در محیط گرید برای اولین بار از یک ساختار درخت وزن‌دار استفاده کرده‌ایم، همچنین در این پایاننامه ما روشی را برای کشف منبع در گرید پیشنهاد دادیم که با استفاده از داده‌هایی که در گره‌های محیط ذخیره می‌کند و با استفاده از وزن‌های درخت قادر است منبع پیشنهادی کاربران را از طریق یک مسیر مستقیم و بدون مراجعه به گره‌های غیر ضروری و اضافی و در یک زمان کوتاه کشف کند.

۶-۲- کارهای آینده

در آینده ما می‌توانیم از جهات بسیاری روش پیشنهادی خودمان را بهبود دهیم. از جمله اینکه اگر در روش پیشنهادی کشف منبع خودمان مسئله هزینه منابع و یا فاصله منبع از یک گره را نیز لحاظ کنیم، الگوریتم کشف منبع می‌تواند تصمیمات بهتر و بهینه‌تری را در انتخاب و کشف منابع برای کاربران اتخاذ کند. از راهکارهای دیگری که می‌توان روش پیشنهادی کشف منبع را بهبود داد ایجاد امکان کشف چندین منبع برای یک کاربر توسط الگوریتم کشف منبع است. یعنی اگر الگوریتم را به گونه‌ای بهبود دهیم که قادر باشد چندین منبع را به صورت همزمان برای کاربر کشف کند خواهیم توانست روش را تا حد بسیار زیادی بهبود دهیم.

www.Prozhe.com

www.Prozhe.com

فهرست منابع

منابع فارسی

۱- فتحی، محمود، ۱۳۸۸، اصول و مفاهیم فن آوری محاسبات گرید، تهران: نوپردازان، ص ۳۳۶.

www.Prozhe.com

www.Prozhe.com

منابع انگلیسی

۱- Al-Dmour, N.A., Teahan, W.J., ۲۰۰۵. Peer-to-Peer protocols for resource discovery in the grid, in: Parallel and Distributed Computing and Networks, Innsbruck, Austria.

۲- Berman, F., et al., ۲۰۰۳. Adaptive computing on the grid using AppLeS, TPDS ۱۴ (۴).

۳- Bharambe, A.R., Agrawal, M., Seshan, S., ۲۰۰۴. Mercury: Supporting scalable multiattribute range queries, in: Proc. of ACM SIGCOMM, pp. ۳۵۳–۳۶۶.

۴- Buyya, Rajkumar.,Venugopal, Srikumar., ۲۰۰۵. A Gentle Introduction to Grid Computing and Technologies, Computer Society of India.

۵- Chang, R.-S., Hu, M.-S., ۲۰۱۰. A resource discovery tree using bitmap for grids. Future Generation Computer Systems. (۲۶) ۲۹-۳۷.

۶- Chien, A., Calder, B., Elbert, S., Bhatia, K., ۲۰۰۳. Entropia: Architecture and performance of an enterprise desktop grid system, J. Parallel Distrib. Comput. ۶۳ (۵).

۷- Foster, I., Kesselman, C., ۱۹۹۷. Globus: A metacomputing infrastructure toolkit, Int. J. High Perform. Comput. Appl, ۲, ۱۱۵–۱۲۸.

۸- Germain, C., Neri, V., Fedak, G., Cappello, F., ۲۰۰۰. XtremWeb: Building an experimental platform for global computing, in: Proc. of IEEE/ACM Grid, December.

www.Prozhe.com

www.Prozhe.com

۹- Karaoglanoglou, K., Karatza, H., ۲۰۰۸. Resource discovery in a dynamical grid system based on re-routing tables, In: Simulation Modelling Practice and Theory, Elsevier, ۱۶(۶), July, pp. ۷۰۴–۷۲۰.

۱۰- Karaoglanoglou, K., Karatza, H., ۲۰۱۱. Resource discovery in a Grid system: Directing requests to trustworthy virtual organizations based on global trust values, The Journal of Systems and Software (۸۴) ۴۶۵–۴۷۸

۱۱- Khanli, L.M., Kargar, S., ۲۰۱۱, FRDT: Footprint Resource Discovery Tree for grids, Future Gener. Comput. Syst. ۲۷, ۱۴۸–۱۵۶.

۱۲- Li, J., ۲۰۱۰. Grid resource discovery based on semantically linked virtual organizations, Future Gener. Comput. Syst. (۲۶) ۳۶۱–۳۷۳.

۱۳- Li, J., Vuong, S., ۲۰۰۵. Semantic overlay network for Grid Resource Discovery, in: Grid Computing Workshop.

۱٤- Marzolla, M., Mordacchini, M., Orlando, S., ۲۰۰۵, Resource discovery in a dynamic environment, in: Proceedings of the ۱۶th International Workshop on Database and Expert Systems Applications, DEXA'۰۵, September ۳_۷, pp. ۳۵۶_۳۶۰.

www.Prozhe.com

www.Prozhe.com

چکیده انگلیسی

Abstract

Today, with the development of grid environments and an increased number of resources and geographical distribution, finding a resource discovery algorithm that can provide a user's required resources in a short time with low traffic, is an important task in resource discovery. Although recently developed approaches remove many of the defects of previous methods such as single points of failure and heavy loads as well as significantly improving the system efficiency, a user's query is again sent to unnecessary paths, so regarding the large number of nodes in grid environments, the system efficiency is reduced due to this additional traffic.

In this thesis, we propose a new algorithm and its use of a weighted tree for resource discovery. We use a bitmap where the number of bit positions is proportional to the number of attributes for each resource. Actually, in every node of the tree there will be a footprint of existence resources in children and the descendant of that node, so when the user's request reaches each node, if the requested resources are available in its children or the descendant, we can directly and without referring to extra nodes and unnecessary traffic reach the node which owns the required resource.

We compare our algorithm with previous algorithms using simulations and results and show that the number of nodes visited in our resource discovery algorithm is less than that for other algorithms, and the difference would be significant with an increase in the number of nodes. Also the cost of update in our proposed algorithm is low.

www.Prozhe.com

www.Prozhe.com

www.Prozhe.com